AFIT/GCS/ENG/93D-14

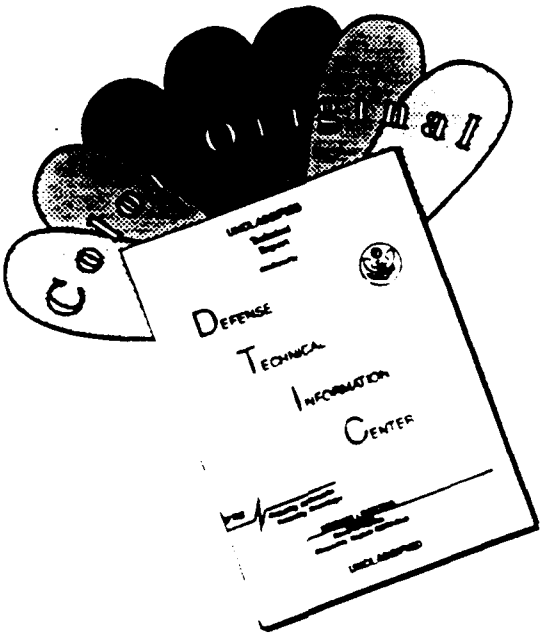
A VIRTUAL ENVIRONMENT FOR SATELLITE MODELING AND ORBITAL ANALYSIS IN A DISTRIBUTED INTERACTIVE SIMULATION

THESIS

Andrea A. Kunz, Capt, USAF

AFIT/GCS/ENG/93D-14

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF COLOR PAGES WHICH DO NOT REPRODUCE LEGIBLY ON BLACK AND WHITE MICROFICHE.

AFIT/GCS/ENG/93D-14

A VIRTUAL ENVIRONMENT FOR SATELLITE MODELING AND ORBITAL ANALYSIS IN A DISTRIBUTED INTERACTIVE SIMULATION

THESIS

Presented to the Faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Computer Science

Andrea A. Kunz

Captain, USAF

December 1993

Preface

I dedicate this thesis to my parents in gratitude for their eternal support, faith and love. I also acknowledge the grace of God for seeing me through this most challenging experience. I owe thanks and appreciation to several people for their assistance and moral support. First, I would like to thank Capt Mark Snyder for his creation of ObjectSim. As difficult and often "unappetizing" as the learning process was, the end result was absolutely spectacular and Satellite Modeler would not have happened without the success of his work. To the VC gurus, Capts Matt "Columbus" Erichsen and Bill "Mickey" Gerhard, I owe a megajillion thanks for the little snippets of code that made my satellites behave properly. Likewise to Maj Mike "Guppy" Gardner -- thanks for leading the way and blazing those trails! Capts Brian "Rm*" Soltz and Kirk "YaKnow" Wilson helped me work through some of the most frustrating moments of my life. I thank them for their friendship, encouragement, borrowed code, and bickering sessions on the night shift. To Capt Alain Jones I offer deepest appreciation for his obsession with photographic perfection. And, even though he wasn't a "graphics weenie", I must thank Capt Vince Droddy. He was there when the going got tough and kept me going.

I would like to acknowledge Lt Col Thomas S. Kelso for his contribution to this entire project. He not only provided me with my first introduction to orbital mechanics, but supplied the code that was the heart of the Satellite Modeler. His analysis and insight were absolutely invaluable. I wish to thank my committee for their guidance and attention throughout this entire project. I am grateful to Lt Col Phil Amburn for his input and objectivity, and to Lt Col Thomas S. Wailes for his positive energy and constant faith in my abilities. And lastly, I thank my thesis advisor, Lt Col Martin R. Stytz for being there when it mattered the most. His enthusiasm for this research was my source of motivation. His understanding and encouragement got me through the program in "virtually" one piece. Thank you sir, for all the fish!

Andrea A. Kunz

Table of Contents

Page

List of Figures

List of Tables

Table Page

Abstract

In response to the need for a more realistic, interactive, and immersive space simulation system, the Joint National Intelligence Defense Staff (JNIDS) has sponsored the development of the Satellite Modeler at the Air Force Institute of Technology (AFIT). The Satellite Modeler (SM) is a virtual environment (VE) application that allows analyst-users to enter a virtual near-Earth space environment and visualize realistic satellite models performing accurate orbital motion. The Satellite Modeler provides manipulation functions that allow a user to interact with multiple satellite models and satellite constellations. The system affords the user multiple vantage points within the environment to view satellites in orbit. It also functions as a network actor in a distributed simulation environment. The Satellite Modeler achieves accurate physical modeling of satellite motion by using the NORAD SGP4 orbital model and associated orbital elements for satellites currently in orbit. System functionality is realized within an object-oriented framework and accessible through a graphical user interface (GUI). This thesis is the second of a three-year effort to create a three-dimensional virtual environment for modeling and manipulating satellite objects.

A VIRTUAL ENVIRONMENT FOR SATELLITE MODELING AND ORBITAL ANALYSIS IN A DISTRIBUTED INTERACTIVE SIMULATION

I. Introduction

1.1 Background

This thesis continues the development of a virtual environment for satellite modeling begun by Capt David L. Pond [Pon92]. The Satellite Modeler (SM) is intended to provide a direct manipulation virtual environment for analyzing and controlling satellite behavior in a distributed simulation. The operation of the Modeler is restricted to the framework of a three-dimensional, graphical model of near-Earth space. Application of virtual environment technology to this complex environment may reduce the cognitive distance separating the ground controller from the spacecraft by placing the user inside a virtual space where visualization and direct interaction can occur. Such an environment can also introduce greater realism and accuracy to distributed simulation by employing satellite data for mission planning, reconnaissance, or navigation purposes.

As the number of organizations and countries launching spacecraft into earth orbit increases, it will become important to be able to visualize several key spatial relationships among these spacecraft. It is much easier to understand these complex behaviors by seeing satellites move and react to external forces than by analyzing raw telemetry data or by performing rigorous mathematical calculations. The ability to display satellite orbits in an interactive three-dimensional simulation can identify potential collisions or passage through perturbing fields before these events can occur. The ability to interact directly with orbiting spacecraft allows a ground-based controller to modify the satellite's orbital parameters and prevent any mishap.

The relationship between one satellite and others in close proximity may be determined by attaching a viewpoint to a base satellite and visually identifying other spacecraft from any point in that satellite's orbit. From the perspective of the orbiting satellite, a user may determine those areas of the earth that are directly visible and covered by a satellite's ground track. Docking maneuvers are more readily understood with a capability to fix a viewpoint at a specific point on one orbiting craft and watching the approaching object.

Visualization from other perspectives may also be useful. For example, ground-based analysts require the capability to determine, for a given set of terrestrial coordinates, which satellites are immediately visible above the horizon. This period of visibility defines a window during which commands may be uploaded to or telemetry data downloaded from the orbiting spacecraft [Dav88]. Likewise, the results of real-time command and control operations may be validated and even predicted through visual confirmation.

Knowing the current status of a satellite's orbit at any instant is critical to mission and reconnaissance planning. It determines when reconnaissance, command transmission, and telemetry data reception can occur. It also provides knowledge of when required communication and navigation satellite resources are available for a planned operation.

Developing an interactive capability that can provide access to this information requires that satellite model and sensor characteristics, orbital mechanics, solar and lunar positional data, and knowledge of the earth's surface features be accounted for. Precise knowledge of a satellite's performance characteristics, orientation, and command set are also important considerations. A virtual environment for satellite activity may be created by applying three-dimensional computer graphics to this physically realistic model. Entry into and interaction with this environment is accomplished by defining visual and direct manipulation interfaces.

1.2 Problem Statement

The control and analysis of orbiting spacecraft requires extensive operator-analyst training. Currently this training does not allow the analyst to have "hands-on" interaction with the satellite because of its impracticality and the high cost associated with error. Normal satellite command and control also does not provide for direct visualization of orbiting spacecraft. A first-person, immersive virtual environment, on the other hand, may be able to efficiently achieve this scenario by placing a controller in a virtual environment that emulates near-earth space and places him in physical control of the spacecraft. The Satellite Modeler was developed to provide this capability. The current Satellite Modeler system is an immersive, direct-manipulation virtual environment that can function in a variety of roles. It can serve as a simulator for training for space systems analysts and operations personnel. It may also be used as an operational tool for real-time command and control of orbiting spacecraft. Since satellite systems do not operate in isolation, the SM functions within the framework of a distributed simulation environment so that satellite data may be incorporated into other interactive simulations and exercises.

1.3 Scope

The problem entailed creating a visually realistic near-earth space environment and accurately modeling the physical behavior of objects therein. To solve this problem within the thesis cycle, I defined it as a collection of specific, achievable objectives. Rather than attempt to model all activities involved with spaceflight, I limited the problem to simulate only on-orbit spacecraft performance. Neither satellite nor ground-based sensor systems were modeled. Furthermore, I did not allow any interactive modification of satellite orbital parameters. To minimize the processing burden, I set an upper limit on the number of satellites rendered in the simulation scene at any one time. I also chose to work with only three different geometric models of spacecraft as these provided sufficient variety for visualization. Viewing satellite

ground coverage was constrained by the classified nature of this data. In order to keep the system unclassified and available for use by more organizations, artificial "look-angle" values were supplied.

The near-earth space environment model included the sun, earth, moon, and stars and defined basic methods for manipulating each according to the physical laws governing their behavior. This model did not account for such astrophysical phenomena as solar wind, Van Allen belts, radiation, or other perturbing forces. This restriction reduced the complexity of the environment and to incorporate the Earth-Centered Intertial (ECI) coordinate system as the basis of my simulation. The lighting model was constrained to only provide illumination from the sun as the primary light source with a minimal contribution by the stars as point light sources. I did not account for the reflection of light from either the earth or the moon, nor did I attempt to model solar or lunar eclipses. With the exception of a simplified umbra, all shadow effects were excluded.

1.4 Assumptions

Development of the Satellite Modeler began with the following set of assumptions. First, I assumed that satellite model geometry would exist in a format that would be compatible with the development environment. I also assumed that I would have access to data that models the behavior of satellites actually in orbit. The NORAD SGP4 orbital model was used, allowing the model to include perturbative effects due to geopotential, atmospheric drag, and solar-lunar gravitation. The earth was assumed to be spherical rather than oblate. Finally, the software to implement the current protocols to transmit data across the Distributed Simulation Internet (DSI) was assumed to exist and be compatible with the application system. This assumption was based on the requirement that the Satellite Modeler function as part of a distributed network simulation.

1.5 Approach

The first step in this thesis effort was a comprehensive survey of research material which covered six major areas: virtual environment (VE) technology, human-computer interface (HCI) design, three-dimensional computer graphics, distributed simulation, orbital mechanics, and space simulation and visualization systems. I began by reviewing the history of VE technology and current research efforts in the field. Issues related to the development of a VE system focused on projecting human presence into simulated realities and the ways in which interfaces could be designed to facilitate this process. Research into the area of user interface design lead to experimentation with several graphical user interface tool kits. This experimentation led to the development of an interface that incorporates many of the HCI design principles I studied. I describe this interface in Chapter 4. The visual aspect of modeling space is the focus of several journal articles that dealt with lighting, texture, and the problems associated with modeling very large environments. Readings in the field of distributed systems, distributed virtual environments in particular, helped determine the feasibility of developing the Satellite Modeler for similar purposes. Prior to reviewing graphically-based space simulations, I studied the fundamental principles of orbital mechanics. Current work in interactive orbital analysis and visualization simulations provided great insight into defining a development approach for the Satellite Modeler. These papers are discussed in Chapter 2.

After completing the initial literature survey, I established the requirements for the Satellite Modeler based on consultations with space systems analysts from the National Air Intelligence Center (NAIC) at Wright-Patterson AFB. I then began development of the interactive interface that would make the required functionality accessible to the user. The application was constructed around the propagation and rendering of satellite models in accurate near-earth orbits. The application system was integrated with the user interface using an object-oriented methodology and system framework. Once the basic system functionality was in place, I expanded the functionality of the system to enable it to communicate with other virtual

environment applications as part of a distributed interactive simulation. Input and control interfaces were added to provide the user access to and immersion in the virtual environment.

1.6 Overview

This thesis describes how the space environment and satellite objects were modeled with the correct physical characteristics and motion as part of an immersive virtual environment. The Literature Review outlines the body of research required to assimilate all the elements to be incorporated into the satellites' virtual environment. It details the issues involved with developing a virtual environment and how three-dimensional computer graphics can be applied to the problem. The review also describes the connection between distributed interactive simulation and virtual environment systems. Finally, it surveys related work on systems that have sought to model the space environment and display orbital information. Chapter 3 focuses on the requirements definition and analysis process and its translation into a system design. The fourth chapter describes how the system components in the design were implemented. Chapter 5 presents the results of this stage of Satellite Modeler development. Recommendations for future work and conclusions are summarized in Chapter 6.

II. Literature Review

2.1 Introduction

This chapter summarizes the literature and work in the five major research areas that are relevant to my thesis effort. Since the primary purpose of this thesis was to create a virtual environment, I completed a survey of the enabling and supporting technologies in that field. Regardless of the terminology involved, the underlying theme of virtual environment technology is the simulation of three dimensional reality in such a way as to immerse the participant completely within the computer generated world. Two of the four remaining topic areas, user interface design and computer graphics, are presented as components of virtual environment system design. The creation of a virtual environment as an end-user system required an examination of user interface design criteria and methodologies. Similarly, to create the environment itself required consideration of computer graphics issues, particularly in the areas of light, shadow, scale, texture, and detail. Finally, simulation in distributed environments and space systems are discussed to show how virtual environment technology may be applied to specific applications. A brief introduction to orbital mechanics is provided to support the discussion of space system applications. This discussion focuses on different approaches to modeling the space environment and the dynamic behavior of spacecraft.

2.2 Virtual Environment Technology

The term "virtual environment" denotes the creation of an artificial world within which real-time, interaction may occur between human participants and the computer-generated reality. In the past, elaborate simulators were constructed to train pilots and astronauts for their respective types of flight and operational control. These simulators were very expensive to build, had to be housed in large facilities, and required many people to operate and maintain them. A virtual environment provides many of the same experiences at a fraction of the resource

expense because it can function as a "simulation in a box". Its existence is defined by the limitations of the computer hardware and software that generate it. In other words, a virtual environment is a collection of sophisticated computer graphics, physical models, and interaction techniques stored in memory that defines some world to be experienced via one or more human sensory systems. The reality of such a world is constrained by the limitations set forth by its supporting technology. The "virtuality" of the environment is the degree to which our physical existence in the real world can be paralleled in the artificial world [Bry92].

2.2.1 Historical Background. Before virtual environment technology existed as a field of research, one had to use one's imagination to gain access to an alternate reality. In his extensive study of virtual environments, Ellis points out that the original architects of virtual reality were cavemen who used pictures painted on cave walls to tell stories of exploits, real and imagined. He also credits fiction writers such as Lewis Carroll with creating alternate worlds for readers to enter through their imagination [Ell92]. With the advent of the twentieth century and the birth of modern aviation came the need to train pilots. This training was one of the first practical applications of simulation because it required an environment for realistic flight experience without expending fuel, aircraft, or maintenance resources, or risking human life. As Marshall notes, however, the first devices supporting the human-machine visual interface were designed without extensive knowledge of human perceptual capabilities [Mar89]. The first computer-based VE interface didn't exist until Ivan Sutherland created a device to present a three-dimensional display of any world that could be graphically represented. His underlying goal was to give the person wearing the device the illusion of actually being in that world [Sut68]. Work in robotics has resulted in the development of artificial appendages and control devices that provide the operator an illusion of grasping and manipulating simulated objects in a virtual environment. Since then, virtual environment research has focused on modeling human psychological and physiological responses. This knowledge of human behavior in the real-world

environment serves as a foundation for the design of computer graphics and user interface devices that provide the same stimuli and feedback in a simulated environment.

2.2.2 Human-Computer Interface Issues. When we attempt to define "presence" or the degree of immersion within an environment, we usually refer to the nature of human physical existence within three-dimensional reality. This existence is confirmed and mediated by the human sensory system. Indeed, it can be argued that there is no existence apart from that which can be sensed. Within the context of a virtual environment, presence is achieved by first creating the visual illusion that what is seen is real. The entire human sensory system is involved in this illusion. If the visual sense can be sufficiently engaged to the extent that the participant believes he or she exists within the framework of that artificial reality, then that person is said to be visually "immersed" in that environment. Other sensory cues also contribute to the illusion. Kalawsky lists the reception of binaural sound among the four basic elements that contribute to immersion [Kal93]. Navigation through and recognition of events within an environment are as dependent upon auditory, and even olfactory, cues as they are upon visual cues ([Leh91], [Mas92]).

Interaction takes place through the haptic and proprioceptive senses. The haptic system processes tactile and force feedback. Tactile feedback is the perception and recognition of a physical object through the skin. Force feedback defines the physical limits that constrain manipulation of that object [Gig93]. Proprioception is the individual awareness of bodily motion and orientation [Kal93]. The primary advantage of this technology is that successful immersion within a simulated environment allows an individual to perform activities with many of the benefits of the actual experience without incurring any of the associated risks or costs. It is for this reason that virtual environment technology is ideal for simulation and training applications.

The visual challenge to virtual environment technology is to create the illusion that what is seen has the same visual characteristics as encountered in everyday experience. Sight is our predominant sense and we use it for identification, orientation, and interaction with other

individuals and objects that coexist with us in our environment ([Nai76], [Sch90a]). Our perception of reality is largely based on the degree to which we perceive dimensionality in our universe. Depth cues such as those listed in Table 2.1 contribute to the human visual experience by providing a basis for orientation and determining spatial relationships in three dimensions. These cues also help us decide what is and is not real by establishing a three-dimensional frame of reference.

Table 2.1 Visual Depth Cues

Type	Cue	Visual Information Provided
Primary	Accommodation	Ability of eyes to focus on an object (lens).
	Convergence	Related to accommodation; physical movement of eyes inward as object approaches.
	Stereopsis	Impression of depth due to physical separation of eyes.
Secondary	Perspective	Relative distance of objects with respect to horizon (linear); relative distance based on diffuse appearance due to atmospheric scattering of light (aerial).
	Size/Height	Perception of relative distance from viewer; objects that are further away appear smaller and closer to the horizon.
	Occlusion	Objects that appear to be closer to the viewer obstruct other objects along the line of sight.
	Texture	An object's texture becomes less dense the closer it is to the viewer.
	Shading/Shadow	Shading aids object recognition; shadows cause objects to either recede or extend forward.
	Luminance	Darker objects are perceived to be further away from the viewer.
	Color	Identification and recognition.

[Sch90a]

In addition to the physiology of vision, the mechanics of vision are important considerations in visual interface design. Factors such as the distance between the eyes, known as inter-pupillary distance (IPD), the distance from the eyes at which visual focus occurs, and the field of view (FOV) must be accounted for if a device is to interface properly to the human visual system. Visual perception is largely dependent upon the quality of image presentation. To be viewable, a display mechanism must have high resolution and an acceptable contrast ratio. Blurring or fuzziness forces an individual to exert unnatural effort to bring an image into focus.

Likewise, a display that is too bright or dim causes fatigue and eye strain. While vision is the key to achieving the illusion of presence in a virtual environment, complete immersion cannot be achieved without the ability to physically interact with one's surroundings.

This visceral interaction is provided by the haptic interface. The haptic sense encompasses both tactile response and manipulation. Just as visual cues are associated with how we make visual identification and association, there are many types of tactile cues that help us identify and manipulate objects. Some of these cues are hardness, texture, weight, shape, size, and temperature. This process is defined by Lederman as "active touch" or "haptic exploration" [Led87]. Devices that either allow an individual to manipulate objects remotely or simulate direct haptic response within a virtual environment must provide the appropriate tactile feedback to the individual. If a person in the virtual environment sees a box, reaches for it, yet receives no signal that contact has been made, then the perception of reality is diminished. Through a series of calculations to determine the configuration and location of a person's hand within the virtual environment and then matching those coordinates to the coordinates of a virtual object, contact can be determined and the appropriate tactile cue fed back into the device. The person in the environment then "feels" the solidity of the object, as well as any auxiliary characteristics such as its texture and shape. The "Sandpaper" system is an example of an attempt to simulate the perception of texture in a virtual environment [Min90]. The complex nature of tactile or surface characteristics is difficult to model; a more achievable goal is that of providing force feedback through the haptic interface [Ber91]. Force feedback alone reports contact between the interface device and an object in the environment by returning an appropriate degree of pressure through the interface to the person manipulating the device [Mas92]. Many of these systems use haptic displays -- displays that show a visual representation of the haptic device as it manipulates objects in the virtual environment -- to provide visual confirmation of contact ([Bro90], [Mas92], [Min90]).

2.2.3 Visual Interface Devices. Visualization devices such as Sutherland's head-mounted display (HMD) were unique in purpose, but as Marshall shows, they were not unique in design [Mar89]. A major goal of the visual interface, as with all interfaces relating to virtual environments, is that its operation be as unobtrusive as possible. The user should not realize that he or she is wearing a piece of equipment or seeing computer generated images instead of real space. Normal visual perception is a passive activity; any exertion necessary for visualizing that departs from this norm disrupts the process. Freedom of movement of both head and body are conducive to unconstrained viewing of one's environment and lead the user to suspend disbelief in the reality of the environment [Kai89]. We will examine visual interface device classification and composition and then show how they are used within the framework of a virtual environment.

2.2.3.1 Classification. Devices that provide a visual interface to a virtual environment may be classified as either wearable or stand-alone. Stand-alone devices, such as Fake Space Labs' Binocular Omni Orientation Monitor (BOOM), permit full range of visualization while remaining physically located in one place [Auk92]. Those devices that are wearable are either worn directly on the user's head without intervening structural support or require some type of supporting head gear on which the various composing elements are mounted. The three types of devices that are generally associated with these two sub-categories are: head-mounted displays, helmet-mounted displays, and Head-Up Displays. The first two are commonly referred to as HMDs and are differentiated by their head mount. The term "head-mounted display" refers to a display device without a structural platform (see Section 2.2.3.2), yet it has become synonymous with all head-worn visual interface devices regardless of their composition or function. The "head-mount" terminology implies that the device is worn directly on the head without the need for protection. The "helmet-mounted display" is less ambiguous; its composition and function are such that it requires the support and stability offered by a helmet-type mount. Head-Up Display (HUD) refers to the technology of see-through display

devices, where computer generated imagery is superimposed on a transparent display, as well as to a specific type of HMD.

2.2.3.2 Composition. The Head-Mounted Display (HMD) superclass encompasses all devices that are worn on the user's head to view a particular environment. The Helmet-Mounted Display is a subclass of Head-Mounted Display; it differs from other HMDs because it requires a durable, stable platform for mounting visual system components. The type of platform classifies the device as either head-mounted or helmet-mounted. Typically, HMD components include a structural platform, display hardware, optics, and a tracking device to determine the user's current viewing parameters.

Structural Platform. The purpose of the structural platform is to allow the user to comfortably wear the HMD [Mar89]. A Head-Mounted Display is worn directly on the user's head with only a minimal support structure for the remaining components. Some head-mounted implementations have used baseball caps to which the viewing apparatus is attached [Pau91]. A helmet mount can be just as ordinary -- bicycle and motorcycle helmets have been used as well as astronaut helmets ([Auk92], [Reb89]). The actual platform is dictated by the components necessary for a given application, the amount of head protection required, and the type of viewing desired. Most HMDs used in virtual environment research use the helmet-mount because of the need to have fast display updates and high-quality imagery, both of which require complex display components that are quite heavy.

Display Hardware. Display hardware is best described as a collection of pieces that comprise the actual viewing apparatus. First, some type of projection device is needed to present an image from its source for viewing. Next, the image must be processed through a series of lenses and possibly mirrors referred to as the display optics. The final image appears before the viewer's eyes projected onto some type of display screen which can take many forms such as a single, flat display or dual, miniature "TV screens". The four basic types of display hardware currently being used for HMD development are the Cathode Ray Tube (CRT), Liquid Crystal

Display (LCD), lead lanthanum zirconate titanate (PLZT) wafers, and light emitting diodes (LEDs). The most common implementations of HMD employ either CRT or LCD technology. Whereas LCDs are small and lightweight, they are, as yet, incapable of providing the high-quality imagery for which CRTs are known. CRTs, on the other hand, are quite heavy and cumbersome when attached to the supporting platform. Despite this drawback, they are the most common display technology used for HMDs due to their capacity for high resolution, brightness, and contrast.

Display Optics. The display optics define the field of view. This is accomplished through the use of a lens structure. The lateral field of view causes us to feel surrounded by what we see. Any time the lateral FOV is limited, a person will experience some degree of disassociation from the real world. Thus, a major requirement for visual perception in virtual environments is that the display provide the widest possible field of view since this is one cue that the user employs to determine if he or she exists "inside" the environment. The wide field-of-view (WFOV) requirement opposes the requirement for high resolution because for a given number of pixels, as the span of viewing increases, the more an image becomes diluted and appears to have less detail. Also, an increase in FOV generally requires larger, heavier optics which, in turn, conflicts with the need to minimize overall HMD weight. It has since become apparent that any unsupported head-worn device cannot significantly exceed a five-pound limit or it will not only disrupt the person's center of gravity but cause fatigue, neck strain, and general discomfort. Just as a person will be less inclined to use something that is too awkward, bulky, or heavy, that person will also tend not to use a thing that is too difficult to use. Ergonomically designed HMDs present features that are easy to identify and manipulate. To tailor the optical fit to each user, an HMD should have adjustments for inter-pupillary distance (IPD), eye relief, and focus (see Figure 2.1). Since the average human suffers from some degree of near or farsightedness, merely having a display that brings the image in up close is not always ideal. The closer the actual display hardware is to the eyes, the smaller the eye relief and the greater the

possibility that someone wearing eyeglasses would be incapable of using such a device. Minimum eye relief should be no less than 30mm; an HMD that allows in/out adjustment of eye relief would accommodate more users, but at a cost of increased optical complexity and overall HMD weight [Str92].

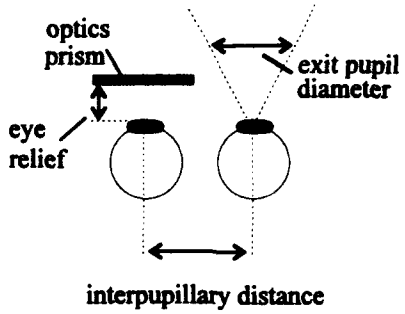


Figure 2.1. Optical Characteristics [Pim93]

Tracking Device. The final component of a typical HMD is the tracking device. The specific scene to be displayed is derived from view-positional data as sampled by a tracking device associated with the HMD. A tracking device consists of two basic parts: a source that transmits positional data and a sensor to detect this data. Basic configurations have the source located on the HMD while the stationary sensor monitors the six degrees of freedom of the source. There are four major categories of tracking technology: electromagnetic, mechanical, ultrasonic, and optical [Pim93]. Electromagnetic devices, such as the Polhemus 3-Space Tracker, conform to the standard source/sensor configuration. The source emits a minimal magnetic field and the sensor processes the location and attitude at the source of the field [Wan90]. These devices are subject to interference from other metallic substances. This interference causes distortion in positional data and imposes a substantial restriction on HMD configuration and usage [Reb89]. Mechanical trackers add a physical linkage between the HMD and tracking control device, usually in the form of a ceiling mounted mechanical arm attached to the user's helmet. Sensors are arrayed at different points along the arm and another sensor is attached to the helmet. Every movement results in a new position and orientation based on

translation and rotation matrix computations. While this method is the most accurate and results in faster response times, it severely limits the motion by the operator [Auk92]. Ultrasonic tracking uses microphones to receive sonic signals and transducers to emit sound waves for position calculation, much like radar, but noise and other interference degrade effectiveness [Pim93]. Optical tracking primarily employs LEDs as the source, while cameras or photodiodes are used for detection. In spite of its susceptibility to interference and relatively slow update rate, electromagnetic tracking is the most commonly used tracking technology in virtual environments due to its being less expensive and more portable than other methods ([Wan90], [War92]). Tracking devices define a separate interface between the display device and the virtual environment control system responsible for processing the position and orientation data. Similarly, tracking technology can also provide information as to the location and orientation of a haptic device within the virtual environment.

2.2.4 Haptic Interface Devices. Just as designers study the human visual system to model visual interface devices, so too is more attention being paid to the haptic system. By understanding how humans grasp and manipulate objects and respond to tactile perceptions, designers can develop devices that provide haptic sensation when interacting with virtual environments. Haptic devices "provide the illusion of manual exploration or manipulation of objects in virtual worlds" and, according to Bishop, are the only interface capable of providing direct interaction with virtual environments [Bis92].

2.2.4.1 Classification. Haptic interface devices fall into two categories. The first type of device is worn on the hand or held. Such a device may serve as a "virtual skin" providing feedback from contact with objects in the environment. It may also enable the person wearing or holding the device to control behavior in the environment through interpretation of hand gesture or motion. The six degrees-of-freedom (DOF) -- position and orientation -- of the hand are constantly tracked for mapping to the virtual environment. Any intersection with an object produces the appropriate tactile response to the haptic device, causing the user to experience the

sensations as though they were caused by actual contact. The second type of haptic device is used for telemanipulation, usually in the form of a robotic appendage or exoskeletal structure that may be worn. Telemanipulation was developed primarily to allow humans to remotely manipulate objects in environments that would be hostile or hazardous, such as in nuclear or waste management facilities. It may also be used in a limited virtual environment to train individuals in the use of complex equipment such as astronaut training or for remote construction of the National Aeronautics and Space Administration's (NASA) Space Station project ([Kal93], [Wic93]).

2.2.4.2 Usage. The ideal haptic device would detect hand motion, position, and orientation, and provide tactile feedback. In the context of a virtual environment, this device enables a person to carry out haptic functions and receive haptic feedback as in the real world. This ideal, however, is currently far from realization.

Like HMDs, a tracking device must be associated with the haptic device. Device configuration typically involves a glove embedded with an array of sensors that transmit positional data to the tracking device and, in turn, back to the system. The VPL DataGlove can sense a variety of human hand gestures through a network of fiber-optic sensors wired into the glove; as the number of sensors in the glove array increases, so does the expected level of gesture recognition [Qua90]. A glove may serve as a control interface to a VE system. For example, an operator wearing the VPL DataGlove may direct the system to stop movement in the virtual world by clenching a fist or begin a flying motion by holding the hand flat and parallel to the ground [Auk92].

One category of device that combines aspects of the glove and telemanipulation devices, such as the Argonne Remote Manipulator, is the master manipulator. The EXOS Dexterous Hand Master (DHM) is a hand-worn structure that resembles a robotic hand skeleton. Its sensors attach directly to the joints of the hand, thus providing greater accuracy in describing motion and

gesture than a glove or robotic hand, but the external skeleton interferes with normal human hand motion [Spe92].

Input devices -- including the classes of glove, multiple DOF mice, and master manipulators -- provide haptic interaction by allowing the participant to grasp, move, or otherwise manipulate objects in the virtual environment ([Auk92], [Iwa90], [Pim93], [War88]). In summary, haptic devices are an important component of transforming the virtual environment interface from passive to immersive. However, before the user can gain entry into the virtual environment, the system must provide a set of controls accessible through the input device(s) that the operator can use to configure and manipulate the environment.

2.3 User Interface Design

The role of the user interface is to provide a link between the user and the computer application. The design of a user interface for a computer system deals with issues ranging from color schemes to the extent to which physical exertion is involved. For instance, a command-line interface requires that a user manually enter arguments and switches to configure the application. A graphical user interface, on the other hand, allows the user to configure the application by selecting various objects. Jacob's direct manipulation user interface represents one end of the design spectrum in its complexity. He defines the interface as a collection of objects that engage in a dialogue with the user as controlled by a User Interface Management System (UIMS) dialogue manager [Jac86]. The UIMS approach provides for centralized control and management of interface activity. Designing interfaces around a central UIMS is becoming more commonplace ([App92], [Lin91], [Lew91]).

The standard keyboard/monitor interface mechanism is no longer the only means for processing dialogue between the user and the application. Audio interfaces allow direct conversation and speech synthesis between user and computer ([Mou93a], [Neg93], [Sch93]). Gesture recognition systems employ multiple DOF input devices that equate motion and

orientation with processing commands ([Bux86], [Kau90], [Kur93], [Qua90]). Gesture recognition also extends into the area of virtual environment interfaces. Immersive environments place the user within the frame of reference of the computer generated world. Interface devices such as the HMD and DataGlove allow the user to interact with that environment in the same manner as in the real world ([Fis93], [Sha92], [Sto91], [Wys93]). The current trend in user interface presentation is away from the command-line and toward graphic metaphors ([Mou93b], [Wal93]) using GUI tool kits such as X-Windows' "Motif" or "Open Look."

As mentioned previously, when designing a user interface for display to a workstation monitor or other viewing device, physical appearance and arrangement of interface objects are important considerations. Color can be used for making associations or coding events. For example, red is commonly associated with hostility or warning. This association may be exploited to signal critical events such as when an error has occurred that requires the user's attention [Sal93]. Effective use of color requires that a variety color be used to enable the user distinguish between the various interface objects and events depicted on the screen. The color mix should also consider the potential for eye strain. Colors of dark or bright intensity are used to direct attention to key areas and should be used sparingly because they tend to produce more eye strain than more subdued shades [Sil87].

In addition to having an effective color scheme, interface objects should be arranged according to the way the user will interact with them as determined by a task analysis. The physical arrangement should facilitate visual separation between objects. It should also serve as a "road map" to guide the user in determining which objects are higher priority and what information is relevant to the current task [Ver93]. Other requirements dictate that task objects with a higher frequency of use be located in a prominent place and that manual data entry by way of the keyboard be used only when necessary.

The user interface design process must start with the user. The user must first communicate his or her requirements to the designer. The designer must then determine from those requirements exactly what the user wants. Finally, it is the responsibility of the designer to translate those requirements into an appropriate user interface [Gou85].

2.4 Three-Dimensional Computer Graphics

Computer generated imagery that defines an alternate reality must possess many of the same qualities as the real-world imagery that surrounds us. In addition to the visual cues mentioned earlier, light, shadow, and shading are very important to modeling realistic environments and objects. Modeling highly detailed, realistic objects and terrain requires consideration of the potential trade-offs between hardware capability and rendering quality. This section will present a brief overview of the techniques and algorithms that may be used to maximize image quality and realism while minimizing the burden on the graphics pipeline.

Light is as important to virtual environments as it is to the real world. Without it, an environment is devoid of color; we would see only the absense of color or blackness. Light is also an important consideration for modeling realistic imagery ([Bro84], [Pou92], [Vin92]). To make the images appear real, illumination and shading models for the environment are required. The ambient, specular, and diffuse components of light each contribute different qualities that produce different shading and reflection effects. For example, highlights on shiny objects are caused by specular reflection; the highlight is the point at which the greatest amount of light is reflected ([Fol90], [Wat92]). The shading and shadowing effects produced by these light sources must also be taken into consideration.

The manner in which an object is shaded affects how realistic it will appear when illuminated [Gar93a]. The shading model applied to a particular object must, therefore, consider the nature of the lighting model. Due to the fact that it is computationally expensive, the Phong shading model is generally bypassed in favor of the more computationally economical Gouraud

shading model ([Fol90], [Wat92]). This is a common trade-off in creating large, complex virtual environments where other elements may also be computationally intensive [Pou92]. Visualization in a virtual environment may occur at distances that can accommodate shading and lighting models. However, if closer examination of an object is desired, then the imagery should be generated so that the image quality improves. The techniques of progressive or adaptive refinement may be applied to adapt the level of image detail and shading consistency to either the viewing distance or to the duration of time that the view is focused on a given object. Bergman's adaptive refinement technique works by increasing the complexity of the rendering scheme over time [Ber86].

In addition to shading, application of texture to polygonal representations of objects can dramatically increase the quality of their appearance ([Fol90], [Pea85], [Wat92]). Textures may be generated by digitizing or scanning a two-dimensional image. These texture "maps" may then be applied to graphical representations of objects by applying a three-dimensional coordinate transformation and then a mathematical function that maps the texture to the specified coordinates. Applying texture to simple or complex surfaces produces high-quality, realistic computer-generated images at minimal cost ([Fol90], [Hec86], [Pea85], [Vin92]).

The next step in creating realistic looking images is to ensure that they are three dimensional. Following from the discussions of light and shading is the topic of shadow generation. There has been a great deal of research into shadow generation algorithms ([Bro84], [Fol90], [Wat92], [Woo90]), but all agree that shadows are a major factor in creating the illusion of reality. The complexity of shadow algorithms inserts yet another cost factor to be weighed against the degree of desired realism. In certain applications, the presence or absence of shadows may be critical.

2.5 Distributed Simulation

Interactive distributed simulation makes virtual environments accessible to multiple applications and participants in real time. Distributed simulation and virtual environments give a user a sense of presence in an environment and allow others to use the data produced by one virtual environment simulation system in other virtual environment-based simulations connected to the same network. Distributed interactive simulation technology uses heterogeneous hosts and a common synthetic environment definition to insert a wide variety of both human- and computer-controlled actors into a single, shared synthetic environment. This is called the environment distribution approach.

The environment distribution approach uses networked virtual environment stations (using long-haul and/or local connections) to form a single environment wherein each node has its own local model of the environment and there are no clients or servers [Tho88]. Presently, the hosts are connected using T1 data links and use a common simulation and network protocol to communicate (Figure 2.2). The protocols currently in use are DIS and SIMNET (see [BBN92], [Bla93], [Har91], [McD90], [McD91], [Mil88], and IEEE STD-1278-93). To reduce network traffic to manageable levels, each actor informs all the hosts of the appropriate dead-reckoning algorithm to use to predict its motion between broadcasts. Each distributed simulation host node broadcasts the significant changes in its actor's state to all the other nodes, thereby allowing the participants to interact at a distance and to maintain a model of the distributed virtual environment that is accurate. Each distributed simulation participant, or host, has the same terrain description, the same geometric description for the actors in the simulation, the dead reckoning model used by each of the other actors, and identification for the actors involved in the simulation. To accurately maintain the state of the simulation, each host knows the velocity and position of the other actors.

Applications for distributed virtual environments run the gamut from interactive role-playing games to military wargaming [Wex93]. SIMNET and NPSNET are examples of

networks dedicated to distributed simulation for military applications ([Wex93], [Zyd92]). As more "players" are identified to participate in a distributed virtual environment, the need for processing power increases. Centralized client-server architectures suffer when there is a great deal of communciations traffic across the network [Pim93]. Operating systems that support distributed environments are being developed using parallel architectures to resolve the issues of processing and computational complexity [Gri93].

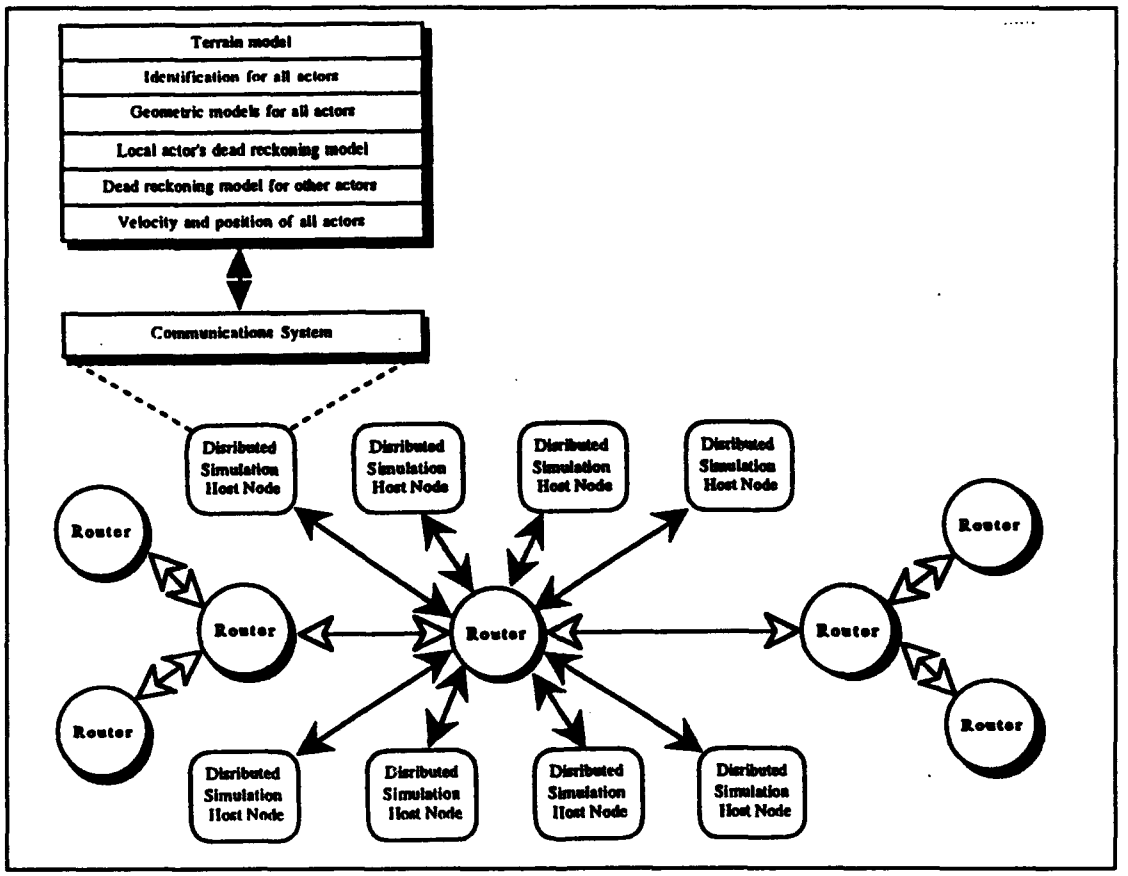


Figure 2.2. Environment Distribution Approach for a Distributed Synthetic Environment

2.6 Space Visualization and Simulation

This section presents an overview of current research in the field of space visualization and simulation. The discussion of three-dimensional computer graphics and VE technology applied to visualizing and manipulating objects in a simulated space environment is prefaced by a review of basic orbital mechanics concepts. These concepts are presented in the context of the NORAD orbit propagation models.

2.6.1 Orbital Mechanics. Representing orbital data in a three-dimensional coordinate system requires an understanding of orbital mechanics. Kepler showed that the motion of a body in orbit about a planet can be described using six orbital elements (Figure 2.3). These "classical"

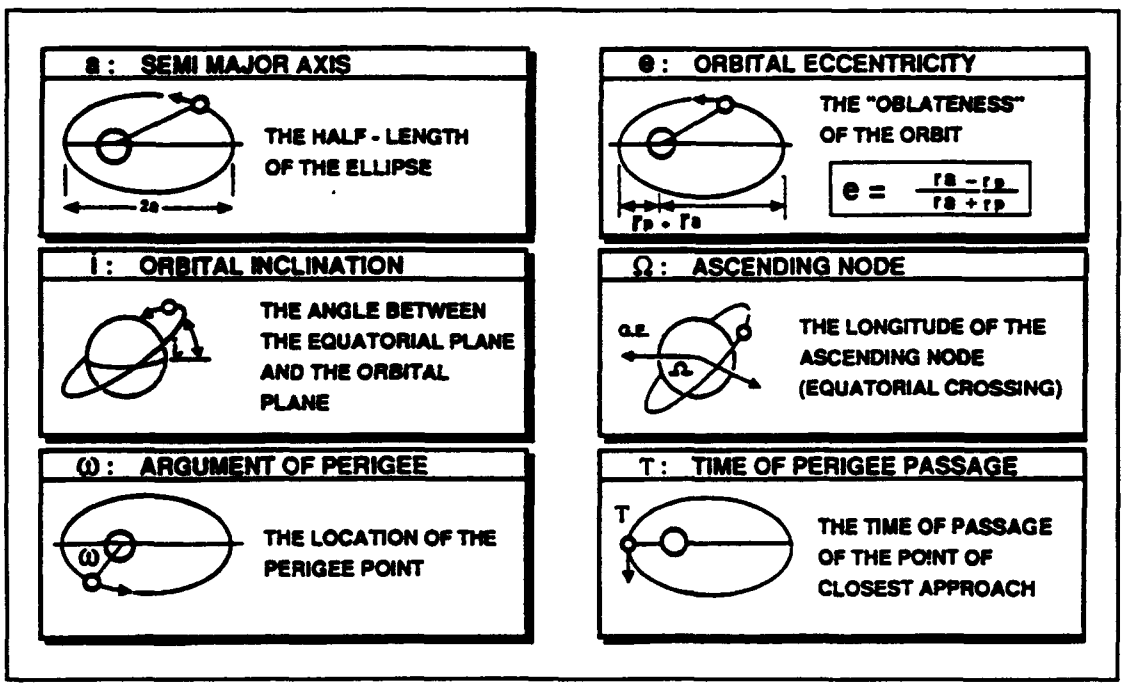


Figure 2.3. The Six Keplerian Orbital Elements [Log92]

orbital elements are defined as follows. The semi-major axis (a) represents the size of the satellite's orbit. Orbital eccentricity (e) describes the shape of the orbit and is equal to $(r_a-r_p)/(r_a+r_p)$. The values r_a and r_p represent the radius of the apogee and perigee, respectively. The angle between the equatorial and orbital planes is the inclination of the orbit (i). The ascending node (Ω) specifies the

right ascension at which the satellite's orbit crosses the equatorial plane. The argument of perigee (ω) specifies the angle between the ascending node and the point at which the satellite's orbit comes closest to the earth. The time of perigee passage (τ) is the time at which perigee occurs ([Bat71], [Dan88], [Log92], [Rim89]).

The North American Aerospace Defense (NORAD) Command's Space Surveillance Center (SSC) maintains a database of spacecraft element sets based on a mean element set prediction model. Of the five models described in [Hoo80], the SGP4 and SDP4 models are currently used to generate orbital elements. The SGP4 model is used for computing near earth (time to complete a single orbit, or period, of less than 225 minutes) orbits. This model factors in the secular and periodic effects of the earth's gravity and atmosphere. The SDP4 model computes deep-space (period greater than or equal to 225 minutes) orbits that account for solar and lunar gravitational effects [Hoo80].

The NORAD elements (Figure 2.4) contain four of the six "classical" orbital elements and are used in the propagation model's calculations for predicting satellite orbits. The NORAD two-line element set substitutes mean motion and mean anomaly for the semi-major axis and time of perigee passage, respectively. Kepler defined mean motion to be

$$n = \sqrt{\frac{\mu}{a^3}} \tag{2.1}$$

where a is the semi-major axis and μ is defined as

$$\mu = GM \tag{2.2}$$

where G is the universal gravitational constant equivalent to $6.67 \bullet 10^{-11} \frac{m^3}{kg\, s^2}$

and M *is the mass of the earth* $\approx 5.98 \times 10^{24}\ kg$

Mean anomaly may then be defined in terms of mean motion as $M = n(t - \tau)$ where t is the current time and τ is the time at which perigee occurs ([Bat71], [Roy91]). Refer to [Hoo80] for a complete description of the NORAD orbit propagation models.

card #	satellite number	class	international designator	epoch time	rev/day^2	rev/day^3	bstar	ephemeris type	element number
LINE 1									
card #	satellite number		inclination (i)	right ascension of node (Ω)	eccentricity (ϵ)	argument of perigee (ω)	mean anomaly	mean motion	epoch rev
LINE 2									

Figure 2.4. NORAD Two Line Element Set [Hoo80]

The "two-body" problem in Physics defines a closed system where both of the bodies are assumed to be point masses and the only forces in the system are those gravitational forces acting along the vector between their respective centers of mass [Bat71]. The NORAD models are based on the assumptions that the earth is not a perfect sphere and that other perturbative forces of a non-uniform geopotential exist. The position and velocity vectors derived from the propagation model are generated for the Earth-Centered Inertial (ECI) coordinate system [Hoo80]. The ECI model places the center of mass of the earth at the origin of a rectangular coordinate system; this simplifies graphical representation and visualization [Oca90]. All coordinates represent fixed locations in space; the coordinate system remains inertial and the earth is allowed to rotate on its axis. Given that the earth remains at a fixed location in the coordinate system, the motion of all other objects is defined with respect to the inertial earth.

2.6.2 Application Systems. Several systems have been developed to aid in the analysis and visualization of spacecraft data as well as provide the means for interacting with orbiting spacecraft ([Eyl87], [Fen91], [Hag86], [Hal87], [Hit92], [Kob73], [Mar93], [Oca90], [Smi83]). The depiction of satellite orbital motion forms the basis for a large percentage of these systems because a three-dimensional, graphical representation provides an important visual aid to understanding complex orbital maneuvers and motion. Virtual environments for space applications are primarily designed to simulate a first person presence in space, but are also used to train ground-based operators and to simulate mission planning and launch activities [Smi83].

Visualizing how an object in geocentric orbit reacts to changes in its orbital parameters was the subject of work done at the University of Kansas by Cesar Ocampo. His system modeled the space

environment in two and three dimensions. Both models employed an ECI coordinate system. Ocampo affirms the strengths of quantifying and graphically representing this information and notes that future work will expand the range of visual interaction [Oca90].

The ability to view the satellite in its environment in real time seems to offer similar advantages to operational analysis and even spacecraft design. Kobayashi and his collaborators developed such a system for the Japanese space program [Kob73]. One of the key design principles of their system was that it perform real-time satellite orbital analysis in a variety of user-defined configurations. Visualization Simulation (VISIM) is another system used for viewing spacecraft simulations. Features that set VISIM apart from other simulations are its abilities to configure more than a single type of spacecraft for a given simulation and to accept real-time, operational telemetry from orbiting craft. It also provides visualization of data and imagery within the spacecraft sensor's FOV [Mar93].

NASA is the major pioneer in the area of space simulation. It sponsored the development of the first simulation that employed computer graphic imagery in the early 1960s. Since then, NASA has been involved with several simulation efforts, including the Virtual Visual Environment Display (VIVED), the Virtual Interactive Environment Workstation (VIEW), the Computer Graphics Pilot Project (CGPP), and the Virtual Planetary Exploration (VPE) Project ([McG93], [Jac92], [Hag86], [Hit92]). VIVED was NASA's first true virtual environment simulation system [McG93]. As VIVED matured, NASA established an entire facility dedicated to development of virtual environment technology in support of space simulation -- the VIEW La. VIEW was chartered as a multi-role laboratory for developing immersive, three-dimensional systems to simulate different events such as telerobotic repair of satellites and extra-vehicular activity (EVA) missions [Jac92]. An individual may enter VIEW and experience one of many different immersive simulations using HMD, glove, and tracking devices. The CGPP is designed to exploit interactive computer graphics capabilities to display orbiting spacecraft in simulated missions. Computer graphics simplify the presentation of an orbital model by replacing complicated textual data with three-dimensional imagery [Hag86]. A direct manipulation GUI further lowers the barriers to interaction by visually organizing

data access and manipulation features to provide the user-analyst direct control over the simulation [Hag86]. VPE is an example of a simulation with a large visual scope; its purpose is to provide visualization of Mars using scientifically accurate representations of its surface [Hit92].

While each of these systems presents information in a graphical format, they are all, with the exception of VIEW, stand-alone, workstation-based simulators. As a result, they do not provide an immersive experience nor do they engage a person's automatic spatial assessment capabilities. To prepare individuals for the physical reality of the space environment, these simulators can be used to train them to function and perform activities given the natural conditions and constraints of space without the associated risks. The NASA Space Station project has promoted research in this area. By applying virtual environment technologies, astronauts, mission specialists, and station operators may be trained in "space simulators," such as VIEW, much like pilots are trained in flight simulators. This capability provides a virtual experience that emulates reality and allows individuals to be more effective and productive once they begin to operate in the real environment ([Fen91], [Jac92]).

2.7 Summary

This chapter presented five topics related to the problem of creating and interacting with virtual environments. Specifically, it addressed how these concepts may be applied to space visualization and simulation for modeling satellite orbital behavior. It is important to first understand the nature of virtual environments and the interfaces that enable humans to interact with them. One must then consider how to present this technology so that its use is well-defined and readily accessible. At the same time, sophisticated computer graphics must be applied to generate an environment that is visually realistic and believable. The space environment presents unique challenges in modeling light, shadow, detail, and object scale. Related work in the areas of orbital analysis, satellite modeling, and distributed simulation provides a functional basis for the design and development of virtual environments.

III. System Requirements and Design

3.1 Introduction

This chapter defines the requirements for the Satellite Modeler and their incorporation into the system design. The requirements process consists of definition and analysis phases. General requirements are introduced in the discussion of the definition phase. Detailed requirements were generated in the analysis phase. The overall requirements formed the basis for system design while the detailed requirements translated into specific system capabilities.

3.2 Requirements

3.2.1 Requirements Definition. An assessment of the first year prototype system combined with specific NAIC and AFIT interests resulted in the definition of four major system requirements. First, the system must accurately model the orbital behavior of spacecraft within a visually realistic space environment. The prototype supported static display of satellite models with dynamic motion achieved through repeated manual placement of objects and recording these manipulations to an output script. Light source selection was the only environmental feature supported. The second requirement was that the system support configuration for, and operation in, both a stand-alone, workstation-based environment, as well as a virtual environment. Third, the system must also include a user interface that provides for interactive configuration of the simulation. Access to, and configuration of, the original implementation was accomplished via a command-line interface. Finally, the system should function as a network player in a real-time, distributed simulation environment. The addition of a network capability would extend the system's functionality by providing other sites with protocol data units (PDUs), formatted into a standard protocol, to realize broader simulation scenarios.

3.2.2 Requirements Analysis. I identified specific system capabilities and their priorities with respect to system design based on my regular meetings with NAIC representatives.

Priorities were assigned according to user interest, complexity, and feasibility. Table 3.1 summarizes the final array of detailed requirements in order of their relative priority. The analysis phase was an iterative process whereby meetings with the user refined the general requirements into specific required functionality. User requirements centered on the need for flexibility, usability, and accuracy. Once the desired system capabilities were established, I began the process of determining how the user wanted to access and manipulate this functionality.

3.3 Design

System design issues were considered as the requirements were refined. Before beginning the actual design of the system, I established a design methodology. The design itself consisted of three identifiable phases. Selection of a target hardware platform was the first step. With an equipment base established, a software base could then be tailored to exploit the hardware capabilities. Software design fell into two parts: the first part focused on designing the user interface with rapid prototyping; the second half revolved around designing the application itself.

3.3.1 Methodology. I based my approach to system design on an object-oriented methodology. This decision was influenced by the availability of ObjectSim, an object-oriented framework for virtual environment applications [Sny93]. Using this approach, each major system component was defined as an object class. Where possible, inheritance was used to minimize the creation of new classes. Attributes defined the physical characteristics of each object. Methods provided access to, and manipulation of, individual object classes. I designed these methods as component software modules to encapsulate specific operations relevant to the object class. Functionality within modules was decomposed to the lowest level to keep module size at a minimum. This decision was based on the software engineering principle that the

Table 3.1
Satellite Modeler Detailed Requirements

REQUIREMENT AREA	DETAILED REQUIREMENTS
User Interface	Provide interactive configuration of all simulation options.
	Input based on file selection with minimal data entry via keyboard.
	Interactive, on-line help facility.
Hardware Capabilities	Support distributed interactive simulation.
	Accept models in geometry format exportable by the Euclid CAD package.
	Process models averaging 25,000 polygons at a minimum of 12 frames per second with a target of 30 frames per second.
	Provide visual output to HMD, workstation monitor, television, and VHS VCR.
	Input interfaces should include tracking device, Spaceball, mouse, keyboard, and voice control.
Simulation Environment	Visually realistic objects including the earth, stars, and moon.
	Environment objects should behave according to orbital mechanics within an Earth-Centered Inertial coordinate system.
	Positioning and size of objects should be accurate according to scale.
	The sun should be the primary light source with minimal light contributed from reflection and stellar objects.
	Shadows cast based on position and orientation with respect to the sun.
	Support network connection and broadcast of satellite data.
	Provide real-time, system, and user-defined display rates.
	Specify simulation time control parameters.
Satellite Models	Load into simulation without loss of physical characteristics.
	Accurate orbital motion based on orbital mechanics and input element sets.
	Nadir pointing orientation.
	Manipulate satellite components according to their natural physical constraints.
	Satellite models should be scaled to size.
	Model near-earth orbits.
	Configuration based on grouping into constellations.
	Limit total number of satellites in simulation to 25 for initial development.
Configurable Viewpoint	Viewpoints at fixed celestial and terrestrial coordinates.
	Ground-based viewpoint should provide optically enhanced view to simulate viewing through high-powered telescope.
	Identify proper ground coordinates for viewing satellites passing overhead.
	Attach view to any orbiting object in the simulation.
	Define independent orbit for viewpoint.
	Focus view at any fixed location in space, on the earth, or on any satellite object.
	Provide capability to focus viewpoint at specific points on satellite model.
	360 degree range of motion for view in all directions as well as zoom in/out.
	Support interactive configuration and relocation of all viewpoint types.
Visualization Cues	Display earth umbra as a cue for solar direction and intensity.
	Satellite trail to show path and direction of orbit.
	Locator objects to identify and distinguish between satellite constellations when viewing at large distances.
	Satellite ground trace to show coverage of geographic regions and identify communications and data transmission windows.
	Level of detail switching as view approaches or recedes from models.

potential for error within a module is directly related to its size. Reducing the possibility of error during the design phase facilitates software testing and debugging and establishes a quality baseline for future modifications [Pyl91]. Use of modular design not only contributes to extensibility but to programmer understanding and overall system maintainability [Cor89].

3.3.2 Hardware Specification. System hardware requirements called for a graphics workstation capable of rendering up to 25 satellite models averaging 25,000 polygons each at a target rate of 30 frames per second. In addition to the standard keyboard and mouse interfaces for the stand-alone mode of operation, the system must also accept input from both Spaceball and tracking devices. The system must support video output to a workstation monitor, HMD, and television to record sessions onto videotape. For operating in a virtual environment scenario, the system should accept voice commands for configuration and control or provide for alternate VE configurations.

3.3.3 Software Specification. A major consideration in developing the software design was the availability of the ObjectSim framework. Originally, this version of the Satellite Modeler was to be built upon the prototype system. The advantages offered by ObjectSim outweighed the extensive re-engineering needed to incorporate all the required functionality into the prototype. ObjectSim provided ready-to-use templates for building virtual environment application objects, as well as hardware and software interfaces. ObjectSim was developed on a UNIX platform using the C++ programming language and served as an intermediate interface to the rendering capabilities of Silicon Graphics' Iris Performer.

3.3.3.1 Object Diagram. The object diagram at Figure 3.1 shows how a substantial amount of the overall Satellite Modeler design is reused from the ObjectSim architecture. This top-level diagram also shows the unique object classes identified for the Satellite Modeler application based on the decomposition of system requirements. The five primary objects within the Satellite Modeler simulation are: the User Interface object, the Application object, the Satellite player object, the View player object, and the Space Terrain object.

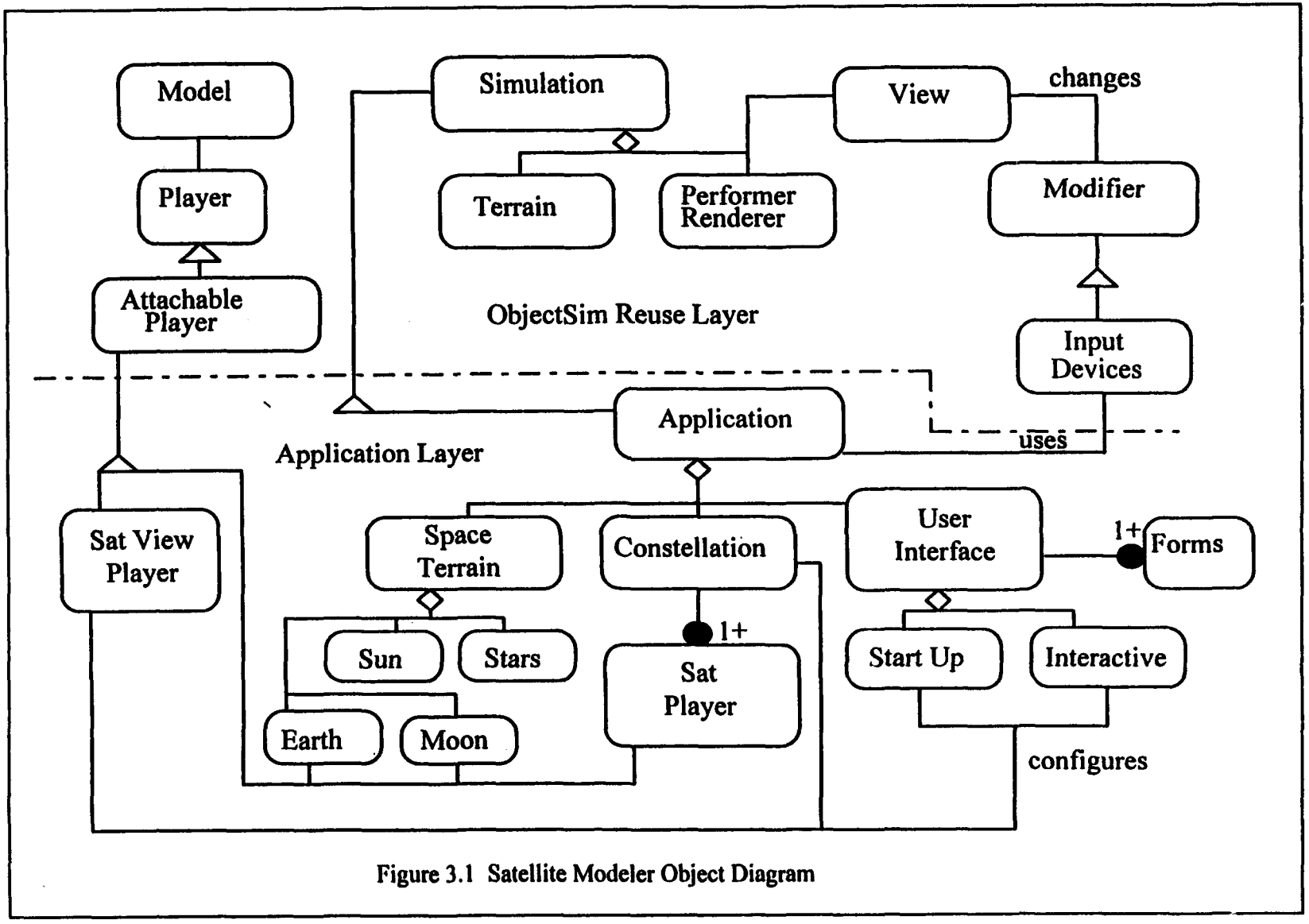


Figure 3.1 Satellite Modeler Object Diagram

ObjectSim defines a variety of different object super classes, among them are the simulation, player, and terrain classes. The Simulation class defines the main control mechanism for ObjectSim applications. Player objects are instantiations of the ObjectSim Player super class and represent entities that exist, change state, and may be propgated in the simulation. Attachable players are instantiations of player objects to which a simulation viewpoint may be attached. All player objects inherit a propagate method from the Player super class. This method allows these object to be manipulated within the simulation. The terrain class provides for tailoring the visual representation of the environment to the unique requirements of the application.

The Application object inherits a set of predefined subordinate objects, attributes, and methods from the ObjectSim Simulation class. As the diagram shows, the Application object controls all other objects in the simulation. Besides the inherited class structure, the Satellite Modeler Application consists of the User Interface, Space Terrain, one or more Satellite Constellations, and a single View player. The Application also inherits a propagate method from the Simulation class; this method controls the propagation of all simulation objects.

The User Interface is an abstract object class that consists of Start-Up and Interactive interface entities. The User Interface is responsible for managing the screen interface through which the user communicates configuration and processing requests. Each satellite instance is configured through the User Interface which is controlled by the Application.

The Satellite player object represents the central focus of the system. Individual satellite objects belong to the Constellation super class. This aspect of the design fulfills the requirement that satellites be grouped into constellations. Given the requirement for attaching the viewpoint to an orbiting satellite and the need for defining other types of views, both the Satellite and View player objects are defined as instances of the Attachable Player super class. The Satellite View player object class encapsulates the essential characteristics and capabilities associated with all

types of viewing for the simulation. All aspects of simulation viewing are directly configurable and adjustable from the User Interface.

The Space Terrain object is a specialized instantiation of the ObjectSim Terrain class. Satellite Modeler is unique in that, compared to other "normal" virtual environment applications, it does not model an earth-sky environment. The sun and stars are non-player instances of the terrain object; they are static objects in the simulation. The earth and moon are defined as attachableplayers to provide further flexibility for viewing from different vantage points within the sim ation.

3.3.3.2 Class Descriptions. This section will describe the attributes and methods associated with each of the major object classes in the Satellite Modeler. Refer to [Sny93] for an in-depth discussion of the ObjectSim architecture.

Application. As shown in the object diagram, the Application object has direct or indirect control over all other objects in the simulation. Figure 3.2 shows the class description for the Application object.

Application
instance of
Simulation
controls
User Interface
Constellation
Space Terrain

attributes
SimulationState

methods
initialize
propagate
draw

Figure 3.2. Application Class Description

The simulation state attribute maintains the comprehensive state of the entire application at all times. This state attribute will be used to communicate information between objects as a

line of control from the Application to those objects. Figure 3.3 shows these lines of control and communication. Internally, the simulation state determines the ability of the Application to invoke its class methods. Upon system startup, the Application will invoke the initialize method which in turn initializes all other objects in the simulation. When each subordinate object has completed initialization, it communicates a "ready" state to the Application. The line of communication between the Application and the User Interface is set to delay further activity by the Application until the user provides input as to the next state of the simulation. This input may be a request to run, modify, or quit the simulation. When the state changes from initialization to execution, the Application invokes the propagate method which, in turn, causes each of the player objects to invoke their respective propagate methods. Propagation continues until the user inputs a request to either restart or exit from the simulation.

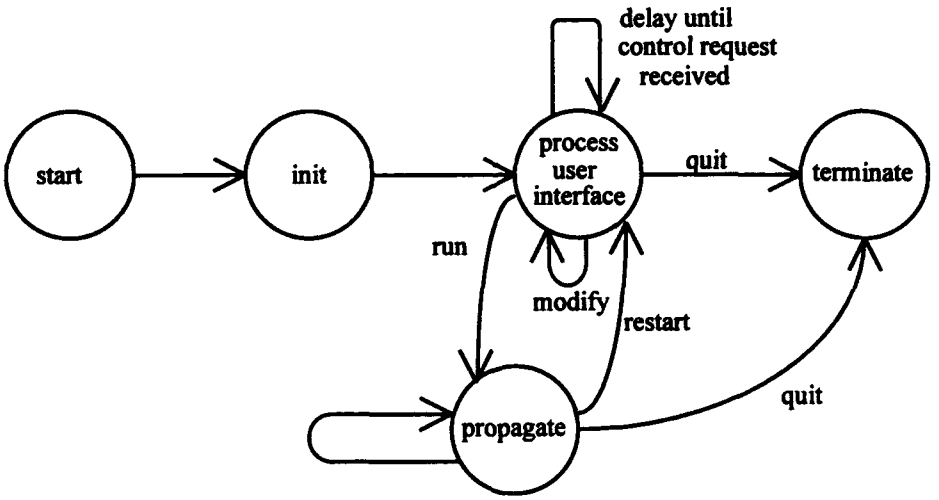


Figure 3.3. Application State Diagram

User Interface. Figure 3.4 shows that the User Interface object is an aggregate of the Start-Up and Interactive interface objects. Together, these two subordinate objects provide configuration control over both the Constellation and View objects. The User Interface as a

whole is a collection of graphical screen forms. These forms, in turn, are collections of graphical interface objects through which the user provides input to the Application. During Application initialization, all interface objects are set to an initial state. As the user processes a dialogue with the interface, the input data are communicated by the interface to the Application. Until the simulation is terminated, the User Interface is active and either communicating a change in state to the Application or configuring the simulation; both activities are driven by user input.

User Interface
configures
Simulation
Constellation
View

Start-Up Interface
Interactive Interface

attributes
Forms

methods
initialize
save
restore
delay
shutdown
process input
report errors

Figure 3.4. User Interface Class Description

In addition to initialization and termination, User Interface methods process input from the user, report error conditions back to the user, save current interface state, and restore a previously saved state. The delay method performs transition between User Interface configuration and the various Application states.

The two subclasses of interface, Start-Up and Interactive, each process user-interface dialogues for a specific phase of the simulation. During initial configuration or reconfiguration,

the Start-Up interface is active. The Interactive interface provides interactive configuration while the Application is propagating the simulation.

The simulation as a whole is subject to configuration through the User Interface. Each interface object defines a part of the overall simulation. All player objects are either configurable directly or indirectly through the User Interface. Indirect configuration implies that the state of each interface object may influence the state of any player object.

Constellation. The Constellation class is an abstract object that is a logical grouping of Satellite objects. By definition, a satellite constellation is "any collection of similar satellites designed to provide multiple coverage or multiple redundancy" [Log92]. The abstract nature of the Constellation class is evidenced by its lack of associated methods (Figure 3.5).

Constellation

attributes
Name
Country
IdentificationNumber
Satellites
SatelliteModel
ColorCode
VisualStateMatrix

Figure 3.5. Constellation Class Description

Its purpose is to provide for configuration of and simultaneous access to multiple satellites. It also serves as a super class of Satellite in that its attributes are inherited by each Satellite.

Each constellation has a name, country, and identification number to distinguish it from another constellation. A single geometry model is defined for each satellite in the group. The color code identifier associates a unique color with each constellation for visual recognition during simulation execution. Finally, the *VisualStateMatrix* maintains the status of each satellite's visual features.

Satellite. The Satellite player object is the heart of the simulation. Each satellite is related to a constellation and inherits country, color code, and geometry model attributes via this relationship (Figure 3.6). The *OrbitalElementSet* attribute defines the parameters that determine the actual orbit of the satellite.

The visual features associated with each satellite -- locator, trail, and ground trace -- are derived from user visualization requirements. Both locator and trail features employ the color

Satellite
instance of
    Attachable Player
related to
    Constellation

attributes
Name
IdentificationNumber
OrbitalElementSet
Locator
Trail
GroundTrace

methods
initialize
propagate
draw
readmodel
broadcast

Figure 3.6. Satellite Class Description

code inherited from the parent constellation.

Most of the methods defined for the satellite object are inherited from the ObjectSim Attachable Player class. The propagate method performs all necessary calculations to determine a satellite's position and orientation within the simulation coordinate system. The draw method performs the actual rendering of the model in the simulation scene at its specified location and orientation based on the geometry input through the read method. The broadcast method is

responsible for generating satellite-specific data and making it available to other actors in the distributed simulation via the Object Manager facility [Sny93].

View. The Satellite Modeler View player object class inherits most of its functionality from the ObjectSim View class (Figure 3.7). The view player itself is a virtual "camera" within the simulation. It can be placed at any location or oriented in any manner.

Satellite View Player
instance of
Attachable Player

attributes
Type
Location
Orientation

methods
initialize
propagate
draw

Figure 3.7. View Player Class Description

Given the many requirements for flexible viewing within the simulation, the Satellite Modeler supports multiple types of viewpoints. The following types of view are supported, as specified in the detailed requirements: fixed space or ground-based location, independent orbit, and attached to simulation object (earth, moon, or satellite). The view type attribute is based on user configuration or reconfiguration of the simulation view. At any given point while the simulation is active, the viewpoint may be changed from one type to another. As part of its simulation state, the Application monitors the view as it switches between types. The current location and orientation of the viewpoint is maintained by the position coordinates and orientation heading, pitch, and roll (HPR) attributes inherited from the View super class. The view propagate method processes changes in view type, location, and orientation.

Space Terrain. The Space Terrain object class is an aggregate of several objects (Figure 3.8). Each terrain object inherits an initialize method. The earth is defined as the root of the simulation scene. Its location is fixed at the origin of the coordinate system. The earth shadow or umbra is defined as a separate player due to the fact that its position must change as the sun's position changes. This change in position is processed by the orient method. The vector attribute defines a vector extending from the sun to the center of the earth; it is along this vector that the umbra must be located. Earth rotation is accomplished by determining the angle offset of the Greenwich Meridian from the vernal equinox (x-axis in the ECI coordinate system) based on the simulation state with respect to calendar date and time.

Space Terrain
consists of
Sun
Position
JulianDateTime
Stars
CelestialCoordinates
Magnitude
Classification
Color
Earth
instance of Attachable Player
Theta
Umbra
instance of Player
Vector
Moon
instance of Attachable Player
Position

methods
initialize
propagate
draw

Figure 3.8. Space Terrain Class Description

One aspect of simulation configuration requires that the user input a simulation date and time. These values are used as a base state for the simulation; all orbital calculations, rotations,

sun and earth umbra placement use this state to establish visual accuracy for the space terrain model. The moon is propagated in a nearly circular orbit and inclined relative to the earth.

Static terrain objects such as the sun and stars are not propagated. The sun's position is calculated relative to the earth based on the calendar date and time of the simulation state. The position of each star is calculated based on its true celestial position converted into rectangular coordinates for display in the simulation scene.

3.3.4 Data Structures. Two major data structures are needed as primary support for the simulation. First, the complex nature of the simulation state requires a structure within which all of its respective components can be maintained (Figure 3.9). This structure keeps track

```
structure
{
    System Level State
        SystemResetFlag
        CurrentInterfaceForm
        ContinuePropagation
        CurrentConstellation
        CurrentSatellite
    Current Object State
        MoonLocation
        SunLocation
    View State
        CurrentType
        CurrentLocation
        CurrentOrientation
        CurrentViewFocus
        ViewAdjustmentParameters
    Visual Feature State
        SatelliteGroundTrace
        SatelliteTrail
        SatelliteLocator
        NewColor
        NewTrailLength
} Shared;
```

Figure 3.9. Satellite Modeler Shared Structure

of current object states and informs the simulation when any change in these states occurs. While read access to this structure may be given to any process in the simulation, write access must be restricted to individual objects to prevent the occurrence of an invalid simulation state. The second major structure serves as temporary storage for the results of dialogue exchanged between the user and the User Interface (Figure 3.10). This structure may only be updated by

structure
{
    Simulation Control
        ResetSimulation
        SimulationMode
        ConfigurationMode
        TotalConstellations
        TotalSatellites
        CurrentDrawStatus
        NewSatelliteModel
        ConstellationArray
    Simulation Time
        TimeMode
        SimulationJulianTime
        PropagationDeltaTime
        ClockTime
        TimeScaleFactor
    Simulation Environment
        Type
        InputDevice
        Visual Feature Selection
            Starfield
            EarthShadow
            Moon
            SatelliteTrail
            SatelliteGroundTrace
            SatelliteLocator
            SatelliteMissionStatus
        Viewpoint Selection
            ViewType
            ViewCoordinates
            ViewDirection
} UserInterface;

Figure 3.10.Satellite Modeler User Interface Structure

methods associated with the User Interface class. The User Interface interprets this raw data and relays it to the simulation which then updates the state values stored in the shared structure.

While update access to these structures is restricted, data retrieval access is permitted for all objects in the simulation by defining these structures globally. This definition facilitates the overall extensibility of the system. Modules added in the future will have ready access to vital simulation data if needed. Additional data structures defined internal to simulation objects should exploit the object-oriented principle of data hiding. Attributes of an object class should be modifiable only through the private methods of that class.

3.4 Summary

The advantages and disadvantages of continuing development using the previous system prototype versus developing an entirely new system were considered early in the design phase. I based my decision to design a new system prototype using ObjectSim on its potential to provide flexible, immediate access to a wider range of required functionality than did the first prototype.

As stated in this chapter, object-oriented principles combined with maximum reuse of existing functionality facilitated rapid design prototyping. Modular design allowed specific functionality to be encapsulated at the lowest level of system hierarchy. Information hiding and restricted access to application and interface data structures served to prevent the system from entering an invalid execution state. Use of this structured design platform simplified the implementation process and laid the foundation to extend system functionality in the future.

IV. System Implementation

4.1 Introduction

This chapter describes how I implemented the Satellite Modeler based on the system design defined in Chapter 3. The implementation approach follows the same principles established for the design methodology. It addresses the implementation of critical functionality. Challenges and departures from the original design are also discussed.

4.2 System Hardware

According to the design phases previously mentioned, the hardware platform was established first. The design decision to use the ObjectSim framework constrained hardware consideration to those systems that run the Iris Performer rendering software. ObjectSim's use of multiple processing "threads" implies an inherent need for a multi-processor machine. This is not a mandatory requirement, however, system performance improves noticeably as the number of processors increases. A three processor machine would dedicate a separate processor to Performer's draw, application, and cull threads. The final target operating environment for the Satellite Modeler is a Silicon Graphics Onyx Workstation with four processors, Reality Engine, and Raster Manager. Both Reality Engine and Raster Manager components are needed to support fast display refresh while handling the exceptionally complex polygonal satellite models. The Onyx provides support for the remaining required hardware interfaces. Included in these interfaces are a high-resolution, stereoscopic Head-Mounted Display with a detachable tracking device. When detached from the HMD, the tracking device can function as a separate input device. Interface support is also provided for a Voice Navigation System running on a Macintosh, Spaceball, and standard keyboard/mouse input devices. To support the distributed simulation requirement, the hardware platform must have connectivity to both local and

distributed networks. The system can also operate on a multiprocessor Silicon Graphics Iris 440/ VGXT Workstation with the same collection of interfaces.

4.3 System Software

4.3.1 Software Platform

Software implementation deals first with the selection of tools and languages used to construct the system. The ObjectSim framework was developed using the C++ language and provides an interface to the Iris Performer rendering software package. Performer's ability to import models in a variety of geometry formats, including the Wavefront object format, satisfied the NAIC's requirement that the system accept models in a format supported by the Euclid CAD package. To ensure the highest possible compatibility between the framework and the application, I chose the C++ language for developing the Satellite Modeler.

Whenever development of a particular feature required functionality that existed elsewhere, I reused software to minimize development and testing time. Nearly all of the orbital methods defining space environment objects and the propagation of satellites were derived from existing source code obtained from Lt Col Thomas S. Kelso in the AFIT Department of Operational Sciences. Collateral routines and functionality were derived from compatible work done by peers. The first system component to be implemented was the User Interface. Application functionality was developed incrementally and added to the software baseline as the interface evolved.

4.3.2 User Interface. Implementing the User Interface required selection of a GUI Toolkit. Of the three packages available -- X-Windows' Motif, the University of Virginia's Simple User Interface Toolkit (SUIT), or the Forms Toolkit -- I chose the Forms Toolkit. I based my selection on the suitability of the package to rapid prototyping and automatic code generation. A rapid prototype GUI was necessary due to the time constraints imposed on the development cycle. The ability to rapidly prototype the interface allowed me to develop a visual

representation of the application quickly without a large time investment. It also allowed me to receive immediate feedback from the user and make incremental adjustments to the interface. Forms' C-language implementation, ease of use, minimal learning curve, variety of supported graphical objects, and close relationship with Silicon Graphics hardware and graphics libraries also influenced my decision.

The interface went through several iterations before all required capabilities were defined. Ongoing meetings with analyst representatives helped refine the appearance of the interface and the arrangement of interface objects. The initial interface consisted of a series of graphical screens accessible by selection of "NEXT" and "PREVIOUS" button objects. This approach was too awkward for the user and posed considerable problems in controlling the state of the interface. A modified dialogue structure allows the user to select interface objects such as buttons, counters, or sliders based upon the type of configuration desired. The interface was designed to guide the user through the configuration process while allowing cancellation, exit or modification at any point during the process. Error reporting and on-line help facilities enable the user to interactively resolve many questions with minimal disruption. All graphical screens for the User Interface are provided in the Satellite Modeler User's Manual.

Access to the on-line help facility is provided on each primary interface screen. Selecting the help button object displays help text relating to the specific screen from which the help function was invoked. Any help topic can then be selected for display. Help scripts are written as a collection of text strings that can be modified as the application changes.

The majority of system configuration is based upon file selection to provide the user flexibility in tailoring the data to the specific task. File selection employs both the file selector and "browser" objects from the Forms Toolkit. The file selector itself allows data to be located anywhere in the system's directory structure; the user can navigate through the structure by selecting different levels or directories. The "browser" object is used to display file contents as well as allow selection of single or multiple files of different types. Where appropriate, options

supported by file selection or "browser" objects also provide data entry fields where the user can manually input data values if so desired.

A special type of button object, the "push" button, was used extensively for configuration items. This type of object provides visual confirmation of specific configuration selections. In cases where there are several configuration options to choose from but only one may be selected at any one time, the "radio" push button is used. This prevents an invalid simulation state. Visual cues were also derived from the "light" button object. Selection was indicated by the change of the button's light field from pale green to a bright yellow.

4.3.2.1 Start-Up Interface. The Start-Up interface consists of the main configuration screen and several supplementary screens that encapsulate all interface objects required to configure the simulation. Interface objects on the main configuration screen are arranged to guide the user in configuring the simulation in a left-right, top-down approach. The far right area of the main interface screen was reserved for control options such as "help", "run", "save", and "quit". The upper left area contains objects for saving and recalling predefined default values. The center area provides access to simulation configuration. The "playback" and "interactive" interface objects define the two modes for the simulation. Selecting the "interactive" option begins the configuration process (Figure 4.1). A completed interactive session may be saved to a session file and played back at a later time. The lower half of the main screen defines options for visualizing the simulation.

The remaining screens for the Start Up interface process user requests to configure the control parameters, space environment, constellations and their respective satellite objects, and the viewpoint for the simulation. The two primary control parameters are the simulation time mode and a user-defined simulation start date-time and time step. The start date and time establish the Julian date for the simulation. Julian time is required for the propagation of satellite orbits and for the correct placement of the sun within the ECI coordinate system. The time step is a value that is used to increment the rate of satellite orbit propagation (see Section 4.4). Three

time modes were defined to fulfill the user requirement for running the simulation at varying rates. The default rate of display renders the simulation as fast as the system can process the images. A user-defined mode allows for interactive control of the display rate; an upper limit is

```
do
{
    process configuration mode
    {
        if PLAYBACK
        {
            select playback file
            load playback file into simulation
        }
        if INTERACTIVE
        {
            configure simulation environment
            configure simulation time
            {
                if file selected
                    read selected time file
                else
                    read keyboard data entry
            }
            configure constellations
            {
                for each constellation
                {
                    select constellation file
                    read input number of satellites
                    select satellite model geometry file
                    for each satellite
                    {
                        select orbital element set
                    }
                }
            }
        configure simulation viewpoint
        }
    }
}
until control option selected
```

Figure 4.1. Satellite Modeler Start-Up Configuration Process

imposed to prevent user disorientation when viewing the images at faster than normal speeds. The time mode also determines the rate of earth rotation, lunar orbit, and satellite orbit.

The visual features of the space environment may be configured from the Start-Up interface. The user may select to have the stars, moon, and earth umbra display upon simulation execution. As Figure 4.1 shows, constellation configuration consists of selecting a constellation input file that contains the constellation name and country identifier. Next, the user selects a single geometry model file to represent each satellite in the constellation. Finally, the user selects a unique element set for each satellite to establish its orbital parameters. This process is repeated for each constellation until all desired constellations have been configured.

The final configuration item in the Start-Up interface is the simulation viewpoint. Each of the two viewpoint classifications consists of two categories (Figure 4.2). Both fixed viewpoints

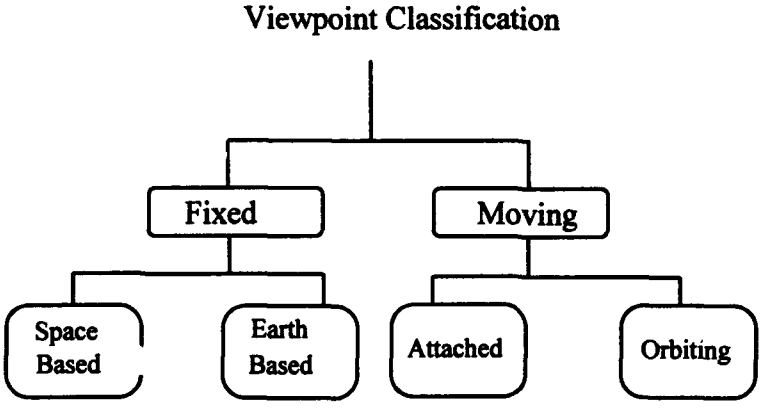


Figure 4.2. Satellite Modeler Viewpoint Classification

require either selection of a viewpoint coordinate file or manual entry of specific terrestrial (latitude-longitude) or celestial coordinates. These coordinates are converted for mapping into the simulation's rectangular coordinate system. It should be noted that the coordinates themselves are fixed but not inertial; the coordinates may be displaced but the coordinate system remains fixed. For example, an earth-based viewpoint will remain fixed at a specific location but that location will move as the earth rotates on its axis. Constellation configuration must be

complete before selecting the attached viewpoint option. The user selects a satellite to attach the view to; a list of all configured satellites is provided at the user's request to facilitate the selection process. Selection of an orbiting viewpoint consists of selecting an orbital element set that defines a spacecraft's orbit. An orbiting viewpoint allows the user to view the simulation as if actually orbiting the earth. All directions of view are focused on the center of the earth. Once the simulation is active, the user may change this default.

After initial configuration is complete, the user selects one of the control options from the right hand area of the main configuration screen. The "run" command activates the simulation and replaces the forms display with the ObjectSim simulation window. Interactive configuration may continue by selecting a function key from the keyboard. It is in the area of interactive configuration that the implementation departs from the original design. For this version of the Satellite Modeler, voice commands were not used because of the potential security risk involved with having speakers and microphones in the NAIC user environment. All user interaction is therefore designed around keyboard, mouse, and Spaceball input devices.

4.3.2.2 Interactive Interface. Whereas the Start-Up interface is a collection of full screen forms, the Interactive interface consists of a series of panels displayed at the bottom of the simulation viewing window. These panels were designed to be unobtrusive yet provide the means for the user to interactively configure the simulation. Like the VISIM and CGPP systems mentioned in Chapter 2, the Satellite Modeler's direct manipulation GUI provides the user with simplified access to complex data.

The Interactive interface provides configuration of the same space environment visual features and viewpoint as the Start-Up Interface, but these actions may occur while the simulation is running. It also enables the user to select satellite visual features by constellation, individual satellites, or all satellites in all configured constellations. The color code defined for each constellation is used to visually identify each satellite as a member of that constellation. Current satellite mission status text -- a dynamic display of key satellite attributes such as

position, velocity, and orientation -- is also color coded to the constellation. Both flight paths and all text displayed to the simulation viewing window are invoked using system Graphics Library (GL) calls rather than Performer functions. Use of the GL calls provided faster and easier access to the desired capabilities while minimizing the computational cost of the code needed to implement these features. The "bubble" objects, used to locate satellites when viewing at great distances, and satellite ground trace "cones" were developed using both the MultiGen and Wavefront modeling packages. This redundancy makes it possible to work with whatever model format might be needed. Also, if the system is to be used outside of AFIT, MultiGen models could not be used due to the proprietary nature of the "flight" format.

In addition to providing flexible visualization options, the Interactive interface allows manipulation of individual satellite objects and switching between time modes. Interactive manipulation of satellite objects may be accomplished by adjusting the values of the heading and pitch slider objects. Per user requirements, all satellites are rendered at a default nadir-pointing orientation. This attitude specification aligns the center of the satellite on the vector from its center of mass to the center of the earth. Interactive modification of the heading or pitch components of satellite orientation is not factored in to the satellite's actual orientation vector. The ability to change the satellite's heading and pitch is provided for visualization purposes only.

Multiple time modes provide the user with greater control over how the simulation operates. The user-defined time mode, unique to the Interactive interface, enables the user to speed up the rate of simulation display by a time scale factor. The "speed up" option keeps the viewer from having to wait the full duration of a satellite's orbit (ranging from 90 minutes to 12 hours or more) to see a complete orbit. As the time scale is increased, all objects in the simulation move at correspondingly faster speeds. System default mode allows the simulation scene to be rendered at the frame rate of the hardware; only the time step input during Start-Up configuration is factored in to the rate of display.

The real-time mode is needed for the Satellite Modeler to work in conjunction with other Distributed Simulation Internet actors. Spacecraft must broadcast their current status information across the network in real-world time since the other players operate within their environment in that manner. Also, some degree of time synchronization is necessary to conduct navigation based on GPS data [Log92].

4.4 Simulation Time

The time control parameters discussed in the User Interface section define the operational scope of the simulation. The three simulation time modes -- system default, user-defined, and real-time -- provide flexibility in simulation viewing. These modes also constrain the physical behavior of all objects in the simulation. The mathematical model that defines the behavior of the object incorporates the time scale derived from the time mode. All Satellite Modeler time calculations are based on Julian time. The Julian date-time standard is particularly well-suited to mathematical computation and is required by the satellite orbital propagation model. It is based on calculating the number of days since January 1, 4713 BC. The Julian system eliminates the complicated bookkeeping required to keep track of days in months, leap years, or conversions to other calendar systems [Pas92].

In addition to the time mode, input values for simulation start date-time and time step are factored into simulation control. The simulation start date and time are entered in calendar and 24-hour clock formats to simplify data entry by the user and then converted to Julian time. The time step is the amount of time that the satellite propagator skips in sampling the orbital element set data. This increment reduces the number of satellite positions sampled and is itself a time scale factor. It is also used in propagating the earth's rotation and the lunar orbit.

Another aspect of simulation time control is the capability to pause or resume motion in the environment. The user may interactively select these options to freeze the movement of all player objects. To do this, simulation time must be sampled at the moment pause and resume are

selected. These time stamps allow the Application to position the players at the proper locations for the current simulation time, given the elapsed time spent in paused mode. If the simulation is not in real-time mode, the visual effect is that once resume is selected, satellite objects will "jump" into their correct positions.

4.5 Space Environment Model

Creating a three-dimensional, graphical representation of the space environment posed several challenges. The initial step to create the models themselves required calculation of a geometric scale to use in their construction. Since the earth is defined to be at the origin of the simulation coordinate system, it was used as the basis for deriving an overall scale factor. Defining light sources for the environment required application of Performer methods. Performer's definition of an earth-sky channel for standard visualization was not used in order to achieve a space environment with a black background. The coordinate systems of models created in either the MultiGen or Wavefront packages were transformed for placement within the Performer rendering structure. While the **z** component of the coordinate system remained constant between modeling software and Performer, the **x** and **y** components were swapped.

The next step involved defining the astronomical parameters and mathematical equations for each model. Objects with dynamic characteristics (i.e., subject to change in position or orientation) were propagated by the application during each processing cycle to maintain accurate placement within the environment. The following sections describe how these models were created and the physical methods used to render and manipulate them within the virtual space environment.

4.5.1. Earth. According to the ECI coordinate system, the earth represents the center of the three-dimensional coordinate system. The graphical model for the earth consists of a two-dimensional texture map applied to a sphere. I defined the sphere to the actual dimension of the earth's radius, 6378.14 kilometers. Since the graphics pipeline cannot process numbers larger or

smaller than the maximum/minimum single-precision floating point number, some type of scaling factor was required. I selected a scale factor of 1/10,000 meters (1/10 kilometers). I chose this scale because it was easy to calculate and interpret, and it worked within the scale constraints of the modeling software.

Scale proved to be a major problem in developing all object models. Software Systems' MultiGen modeling software package is limited in its ability to support wide ranges of scale. The same scale factor used for the earth model was applied to the moon and used to calculate position and distance of all simulation objects. Scaling satellite models with respect to the earth, however, resulted in Performer clipping models from the simulation view. For this reason, I defined the satellite models according to their actual dimensions rather than scaling them using the base scale factor.

The earth model contains an attribute for its angular orientation about the z axis. The earth is rotated about its z axis by an angle θ that is the offset from the vernal equinox based on the current sidereal time (Figure 4.3). *Each pass through* the application's propagation method

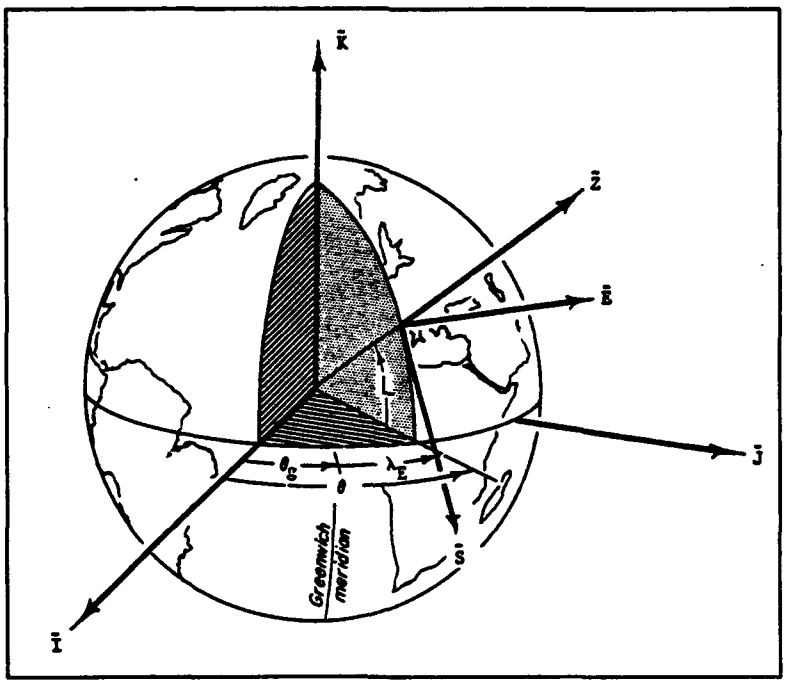


Figure 4.3. Greenwich Sidereal Time [Bat71]

results in an updated earth rotation. Rotation in the real-time mode is subject to the following equations and constant values:

*real time angle of rotation = ((omega_E * degrees per rev)/secday)/frame rate* (4.1)

omega_E is the number of earth revolutions per solar day (≈1.0027379)
degrees per rev is the number of degrees completed in one earth revolution
secday is the number of seconds in a solar day (86,400)
frame rate = 1.0/(current frame time - previous frame time)

The rotation calculation for the system default mode is

system default angle of rotation = ((time step • secmin) • rotation degrees per second) (4.2)

time step is the user input "delta time" value converted to seconds
secmin is the number of seconds in a minute
rotation degrees per second = rotation degrees per solar day/seconds per day

The user defined time mode performs the real-time mode calculation and factors in the currently selected time scale value:

user-defined angle of rotation =real time angle of rotation • time factor (4.4)
time factor = ((time step • seconds in minute) • time scale) (4.3)

time scale is the user selected scale factor (valid range: -5.0 to +5.0)

4.5.2 Sun. The Performer "pfLight" data type defines a primary light source for rendering in a three-dimensional scene. This light source is not defined as a physical model of the sun, rather, it creates illumination associated with a primary light source with configurable location and ambient qualities. The ability to specify a location for the sun is critical to modeling the virtual space environment. Since the ECI coordinate system is used, all objects must move with respect to the earth. Therefore, the sun's position must change according to the Julian date specified by the user for the simulation. The visual effect of the earth rotating around the sun is evident when viewing the location of the sun's illumination upon the earth model.

To create a solar model within the application, I defined a data structure to hold key attributes associated with this object (Figure 4.4). The method to calculate the position of the

sun is invoked during the initialization phase and whenever the simulation date changes. Each time a new position is calculated, it is stored in the *LightSource* data structure. The *NewLocation* flag is set to notify other objects, particularly the earth umbra, of the new position.

```
structure
{
    pfLight* Light;
    vector Position;
    int NewLocation;
} LightSource;
LightSource* Sun;
```

Figure 4.4 Sun Data Structure

4.5.3 Stars. Like the model used for the sun, the stars are rendered using Performer function calls. Using the design specification, the data structure for each star consists of the following elements shown in Figure 4.5. Star objects are rendered at inertial positions in the SM coordinate system because the apparent motion of even the closest star is not detectable given the limits of the virtual environment. Since star objects are static, the Star object class does not contain a propagation method.

```
structure
{
    LightPoint;
    structure
    {
        RightAscension;
        Declination;
        Distance;
    } CelestialCoordinates;
    Magnitude;
    StarClass;
    Color;
} Star;
```

Figure 4.5. Star Data Structure

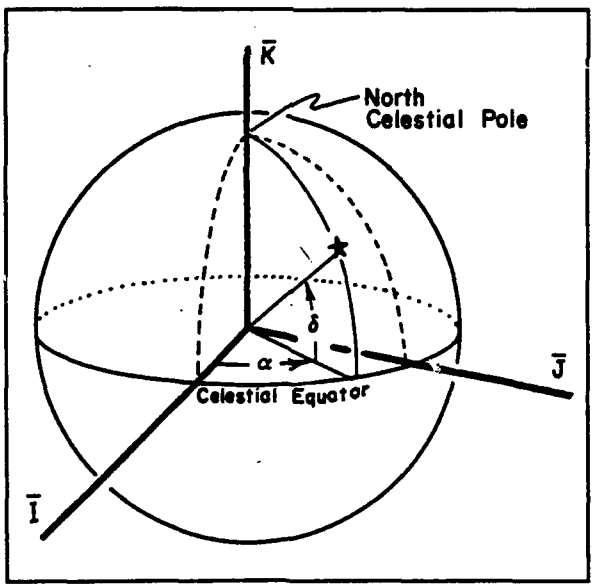
Star objects are defined during initialization based on file input containing approximately 5,000 actual star locations on the celestial sphere (Figure 4.6). Performer functions establish

each star as a point light source with a specific location, size, and color. The equations used to convert the spherical coordinate values into rectangular coordinates for placement within the simulation environment are as follows:

$$x = \rho \cos \alpha \sin \delta \quad (4.5)$$
$$y = \rho \sin \alpha \sin \delta \quad (4.6)$$
$$z = \rho \cos \delta \quad (4.7)$$

ρ is the distance of each star from the origin
α is the star's right ascension on the celestial sphere
δ is the star's declination on the celestial sphere

Each coordinate component is scaled with respect to the earth by the established scale factor. Since the data file only contained the right ascension and declination elements of the celestial coordinates, an arbitrary value for ρ was selected. Using the Performer function to set the scope of the rendering environment, I selected an arbitrarily large value for the distance of each star



α right ascension of star X
δ declination of star X

Figure 4.6. Right Ascension -- Declination Coordinate System [Bat71]

from the earth. To compensate for this contrived distance, I set the screen pixel size of each light point to the minimum value allowed by Performer. This produced a visually realistic starfield.

4.5.4 Moon. The moon is defined as a dynamic player within the simulation because it is continually moving in an orbit around the earth. I modeled the moon as a texture-mapped sphere, with a scaled radius of 173.8 meters. It is propagated in a nearly circular orbit at a scaled mean distance from the earth of 384,400 kilometers. The moon rotates at the same rate as it orbits the earth, so there is no visible rotation to render in the environment.

Given that the moon completes one orbit of the earth in roughly 27.3 days, it is propagated at an angle relative to the center of the earth of ω=.00015 degrees per second ([Bat71], [Dan88]). The following equations were used to derive rectangular coordinates for rendering the lunar model in the simulation:

$$x = r \cos \omega t \tag{4.8}$$

$$y = r \sin \omega t \tag{4.9}$$

$$z = z \tag{4.10}$$

where t is the current time and z is set to zero. System frame rate and time control parameters are accounted for in propagating the lunar orbital model in the same manner as the earth's rotation and the position of the sun. These calculations approximate the moon's actual orbit. It is noted in ([Bat71], [Dan88], [Rim89]) that the orbital elements of the moon are subject to constant perturbation, primarily by the sun. Since modeling the moon was not the primary focus of this effort, I decided to sacrifice accuracy in this area as opposed to that of the satellite model.

4.6 Simulation Viewpoint

The Satellite View Player object facilitates establishment and manipulation of a user's viewpoint within the environment. The view object, like the satellite object, possesses attributes such as position and orientation and can have either an inertial or moving coordinate system. Inertial viewpoints are typically defined by celestial or terrestrial coordinates and may be changed through the interactive view relocate option of the user interface. Earth-based fixed

viewpoints are configured by entry of latitude, longitude, and elevation values. The resulting view will be that of looking through a high-powered telescope out into space. Space-based inertial viewpoints are defined by entry of values for right ascension, declination, and distance from the earth. These celestial coordinates are converted to rectangular coordinates for placement within the ECI coordinate system.

There are also two types of moving viewpoints. The first is a view that is attached to a spacecraft object as it moves in its orbit; this is referred to as "tethered" mode. A tethered view may be attached to any orbiting object in the simulation by specifying a particular satellite constellation and then toggling through each configured satellite. Tethering the viewpoint to the moon or earth is also allowed. The other type of view in this sub-class moves in an orbit that is selected and propagated in the same manner as the orbit of a satellite object. This allows the user to "fly" through the environment in an independent orbit.

If the viewer wishes to change the viewpoint focus to a specific location on a spacecraft, they can do so by selecting the satellite vertex option. A file selector allows the user to choose a file containing vertex information for a particular satellite. These files are constructed off-li. e and the coordinates they contain are positions relative to the center of a satellite model. The coordinates are translated into the simulation coordinate system and the view is redirected toward the new focal point.

The viewpoint may also be focused on any object in the simulation, including the earth and moon. Any object that is the subject of a viewpoint focus is rendered at the center of the simulation window. The default focus for all viewpoints is the center of the earth. While in the interactive interface, the user may adjust the viewpoint to move right or left, up or down, or zoom in or out.

4.7 Satellite Orbital Model

Orbital methods provide for calculation of spacecraft position based on the NORAD mean element set using software routines derived from the SGP4 mathematical model [Hoo80]. These methods calculate the spacecraft epoch from which coordinate position vectors may be propagated for a selected time period. These coordinates are used to place satellite models at their proper location within the simulation's coordinate system. The coordinates define the physical location of the center of the satellite model with respect to the origin of the modeling package. Each component of the coordinate vector is scaled by the simulation scale factor. The satellite model geometry is then rendered at that location according to a nadir-pointing orientation. This orientation is calculated by extending a vector from the center of the satellite model to the center of the earth and then adjusting the heading, pitch, and roll (HPR) of the satellite along that vector.

Propagation of satellite position is performed for each frame of simulation processing. Any visualization features currently selected are also propagated based on the current position and orientation of the satellite. The ground trace "cone" is a semi-transparent unit cone scaled to the distance between the center of the satellite and the center of the earth. The default orientation of the cone is along the same gravity gradient vector as the satellite's orientation. This alignment may be modified based on estimations for satellite look angles for accurate ground coverage. The usefulness of this feature is that it allows the viewer to visually verify windows during which satellite data transmission (upload or download) may occur.

The satellite locator "bubbles" were derived from [Wil93] and may be used as a stand-alone feature to identify individual satellites or constellations when the viewpoint is detached from these models by great distances. Both locator and ground trace features were created as transparent unit models. The locator "bubble" inherits the color code attribute established for the given constellation. This allows the viewer to readily distinguish between constellations. The locator is rendered at the same location as the satellite.

Like the locator "bubble," the satellite trail is colored according to the constellation's color attribute. Trails are rendered using GL line drawing calls and rendered as part of the simulation scene with the "OVERLAY_IN_SCENE" parameter to prevent their obstructing other objects. The data structure for the satellite trail was derived from [Gar93b]. A time delta ratio establishes the frequency with which satellite positions are stored into the trail data structure. A new pixel location is added to the position queue if the time delta has elapsed. The trail is drawn from the satellite's current location out to the oldest pixel location in the circular position queue. The user may adjust the length of satellite trails interactively and may also reset the display of all trails to refresh the simulation window. Resetting the trails clears all pixel locations from the circular queue and the process of adding new locations starts over. The satellite trail is a particularly useful feature in that it not only allows the user to see a visual history of the satellite's orbit, but when the view is attached to a satellite with its trail displayed, the trail indicates the current heading of the satellite in its orbit.

Interactive manipulation of satellites is allowed when the user's viewpoint is tethered to a satellite. The satellite's orientation may be manipulated through the full 360-degree spectrum. As mentioned previously, this manipulation is not factored in to the default nadir-pointing orientation of the satellite. It is provided as a fine adjustment tool for inspecting different aspects of the satellite. Another feature provided for attached viewpoints is a dynamic display of the satellite's current mission statistics including the constellation name, country, satellite name (based on the orbital element set), current position (in kilometers), heading and pitch (the roll component is always zero). When this text display is not activated, a readout of the current constellation and satellite selected for viewing may be displayed. Text is displayed using GL calls to superimpose the text over whatever object may be in the scene. Text is also colored based on the current constellation and/or satellite.

Performer provides the capability to visualize level of detail transition between different resolutions of model geometry. This is done by specifying distances at which each level-of-

detail (LOD) should be switched to the next level. Model resolution information may be modified off-line to configure a variety of visual effects. When satellite geometry is selected for a constellation, the User Interface will read in all resolutions of model that are defined. As the viewpoint approaches a particular satellite model, the detail with which the satellite is rendered will transition from low to high. Likewise, when the viewpoint recedes from the satellite, the detail will gradually degrade. To accomplish this visualization, satellite models must be constructed for each desired level of detail. Low-resolution models are typically composed of a minimal number of polygons and features. As the resolution increases, the complexity of the model also increases.

Several calculations are required to broadcast current satellite status information to other players in an interactive distributed simulation. The position vector must first be converted to meters and then translated from an ECI coordinate system to an Earth Centered Earth Fixed (ECEF) frame of reference [Eri93]. This is done to provide meaningful data to simulation players operating in a round earth environment that does not account for rotation. The translation process involves calculating the number of seconds elapsed since midnight in Julian time

$$time\_elapsed = (fractional\ part\ of\ Julian\ day + 0.5) \bullet seconds\ in\ solar\ day \quad (4.11)$$

then use this value to determine the percentage of one earth rotation that has passed. The angle that the coordinate position must be translated by is given by the following:

$$angle\_rotated = -360.0 \bullet percent\_rotation \quad (4.12)$$

This translated coordinate now represents a valid position within the distributed simulation. The satellite velocity vector is converted to $\frac{m}{s^2}$ and the orientation is converted to radians. The value for satellite acceleration is derived based on the direction of the position vector:

$$acceleration = direction \bullet g \quad (4.13)$$

where

$$g = \frac{\mathbf{GM}}{\mathbf{R}^2} \text{ or } 9.8 \frac{m}{s^2} \tag{4.14}$$

G is the universal gravitational constant and M is the mass of the earth (both are defined in Equation 2.2)
R is the radius of the earth (6378.14 km)

which represents Newton's law of universal gravitation [Zei92]. All of these values are loaded into a satellite information packet structure and broadcast over the network where any active simulation player may receive it. The Satellite Modeler must be configured for a real-time simulation before this information is processed or transmitted.

4.8 Summary

This chapter discussed the implementation of the virtual space environment and how objects were modeled and rendered within the simulation. The focus of this effort was visual realism and accurate modeling of satellite orbital motion. Trade-offs were made in modeling other simulation objects such as the moon and stars to achieve this primary goal. Actual data structures and methods for implementing simulation objects followed the design specification. Additional attributes and methods were added as development progressed to realize greater functionality, efficiency, and accuracy.

V. Results

5.1 Introduction

This chapter discusses the approach taken to ensure that the system, as implemented, fulfills the original requirements. It also describes the process whereby system functionality was validated as performing accurately and according to the design specification. Sample images from validated simulations are presented to show implementation results.

5.2 System Validation

As functional components were tested and added to the software baseline, I conducted system demonstrations for NAIC representatives and AFIT faculty. The feedback I received indicated where specific requirements had been met and where additional work was needed. The user interface provided a graphical representation of the required functionality. Developing the user interface first led to early identification and accommodation of system requirements. Demonstrations were also given to fellow students, project sponsors, members of the Advanced Research Projects Agency (ARPA), and Air Force Space Command (AFSPACECOM) training personnel. Installation of the Satellite Modeler for an ARPA simulation demonstration verified the system's capability to interactively communicate with other simulation participants across a distributed network.

The accuracy of the orbital model was validated by comparing the numerical data produced by the satellite propagation methods with proven results from the SGP4 model. After accounting for the differences in scale, the results produced by the Satellite Modeler for satellite position matched those produced by the SGP4 model running on an IBM 486 PC. Variance in results was accounted for by round-off due to use of double and single-precision floating point calculations on the different platforms.

Visual analysis of satellite propagation for several orbital element sets verified that the satellite models were being rendered in the correct position with respect to the earth. Different time modes

were tested to verify the correspondence between the satellite's orbital period and the earth's rotation. One example that confirmed this is the visualization of a Molniya orbit. From apogee to apogee in its orbit, the earth is expected to rotate 180 degrees since the period of the Molniya is twelve hours. Both earth and satellite models performed as expected. The same held true for a geostationary communications satellite. The Comstar orbit placed the model at a fixed location above the earth and as the earth rotated, it continued to orbit above that position.

The process of repeated demonstration and cooperative analysis uncovered many inaccuracies that were resolved prior to system completion. Items that were not resolved are addressed under "Recommendations for Future Work" in Chapter 6.

5.3 Observations

The visualization of a three-dimensional near-earth space environment forms the basis for the Satellite Modeler's functionality. The visual display is the result of configuring the simulation via the user interface. A detailed description and complete Start-Up and Interactive interface screen displays are provided in the Satellite Modeler User's Manual.

Figures 5.1 and 5.2 show the basic view of the space environment with several visual features selected. In Figure 5.1, the earth is displayed with its umbra aligned in the direction opposite the light from the sun. The moon is visible as a small white sphere. Three constellations are configured, evident by the three different colors of satellite trails. Satellite ground trace cones are displayed indicating the direction and focus of their current ground coverage Figure 5.2 represents a viewpoint in an independent orbit focused on another satellite in the simulation. All satellite visual features are displayed. . Satellite visualization features are depicted in Figures 5.3 through 5.5. The view is attached to a Molniya satellite model in Figure 5.3. The orientation of the satellite and its distance from the north pole indicate that the satellite is approaching apogee. The location of the sun is derived from the visible light and shading of objects in the display. The Interactive interface displayed in the simulation window shows the main control panel used to configure visual features, simulation time,

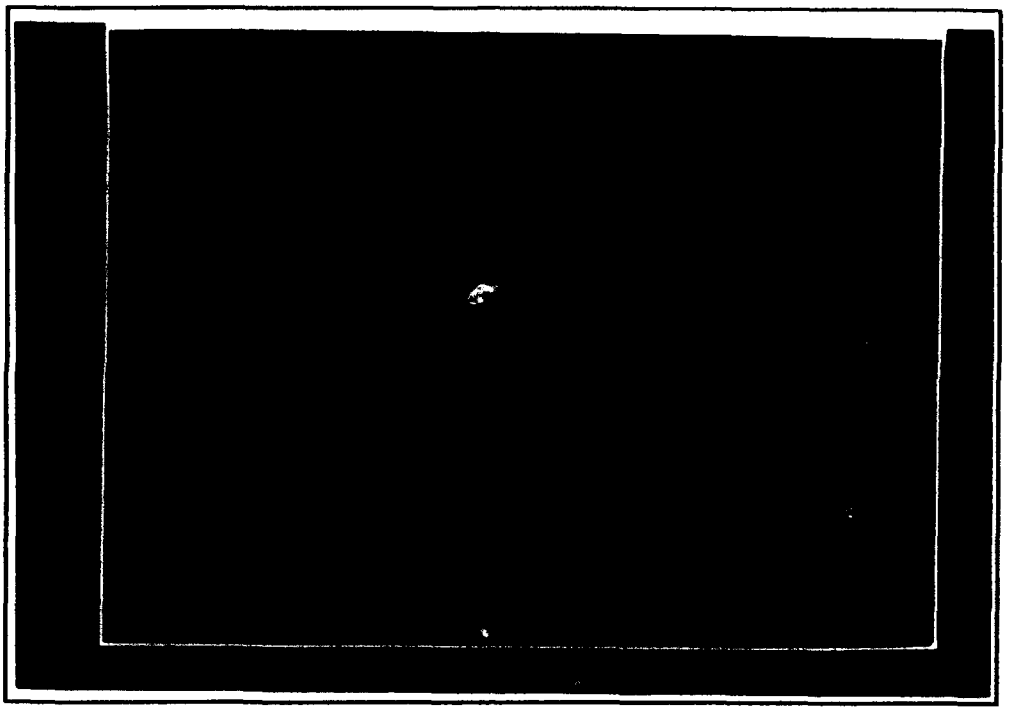


Figure 5.1 Fixed Space Viewpoint
Space Environment Visual Features

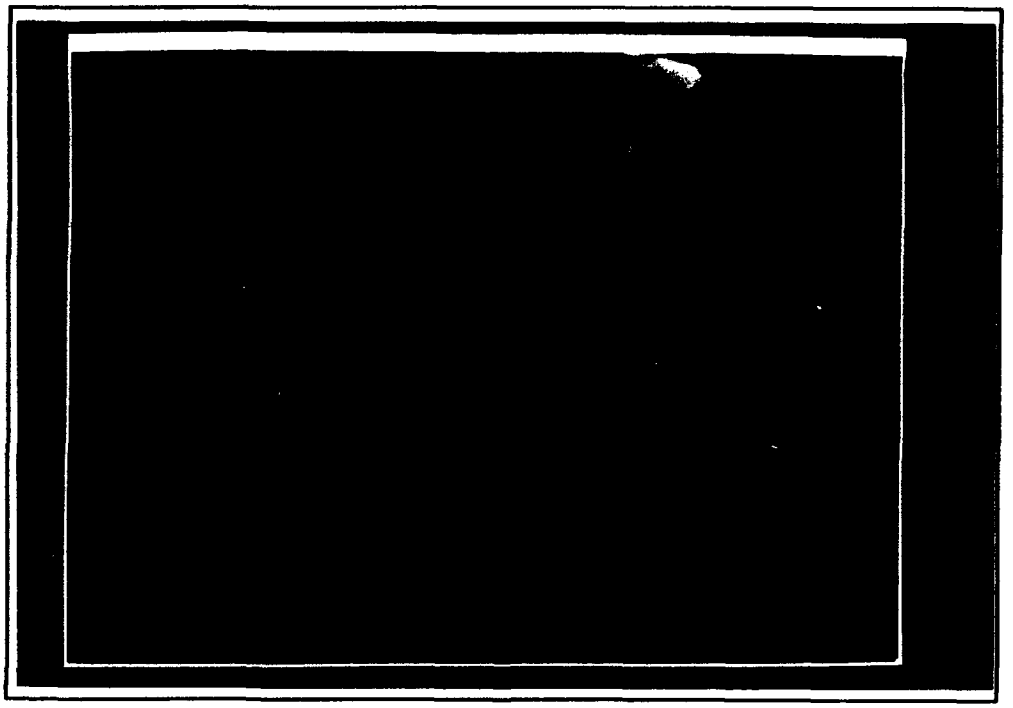


Figure 5.2 Orbiting Viewpoint
Space Environment and Satellite Visual Features

viewpoint, and the space environment. Access to the on-line help facility is also provided from every panel of the Interactive interface. All satellite models depicted in these photographs were created using MultiGen and are constructed to scale.

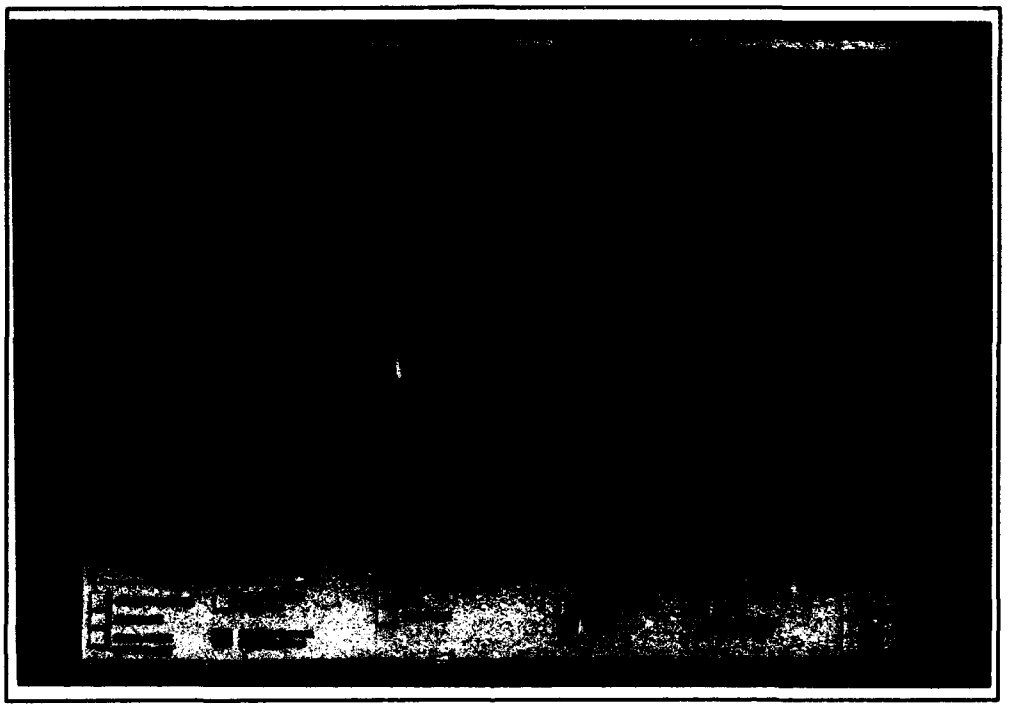


Figure 5.3 Tethered Viewpoint
Molniya Orbit

Figure 5.4 depicts a viewpoint attached to a GPS satellite model. The displayed trail provides an indication that the satellite is proceeding upward in its orbit and the ground trace cone indicates where the satellite currently has visibility over the earth. Two other satellites are also visible in this view. The trail color match is evidence that all three satellites belong to the same constellation. The text display in the upper left hand corner indicates the current constellation and satellite selections used to control viewpoint selection and the display of satellite visual features. Figure 5.5 shows another viewpoint tethered to a GPS satellite model. The satellite locator "bubble" and mission status display are selected. The locator is a unit sphere with diameter scaled to be larger than the largest satellite model (roughly 25-30 meters). The transparency effect allows the satellite model to be seen while the locator is activated. Locator color is inherited from the color attribute associated with the

"parent" constellation. Mission status text is also color coded; it provides a dynamic readout of current satellite entity and orbital parameters. Entity parameters include the name of the constellation

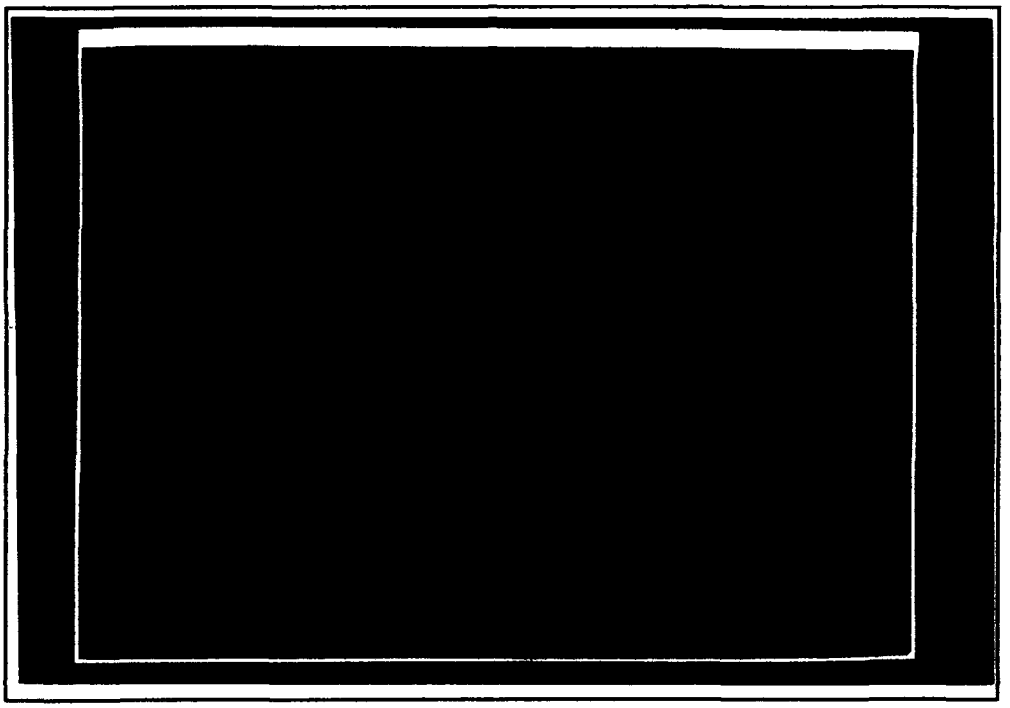


Figure 5.4 Tethered View
GPS Orbit

to which the satellite belongs, the name of the satellite, and the country of origin. Satellite names are derived from the unique orbital element set files used to propagate their orbits. Both constellation name and country of origin data are contained in the constellation file selected by the user during Start-Up configuration. Position, velocity, and orientation vector information are updated for each frame of simulation rendering. When the simulation is paused, indicated by the Time control interface panel, all motion and updates are suspended until the "RESUME" button is selected. The visual results displayed in the photographs above provide a great deal of insight into the capabilities of the Satellite Modeler. However, one must view an active simulation on a workstation monitor, recorded video tape, or through an HMD to appreciate the immersive qualities provided by the visual realism, interaction, and object motion.

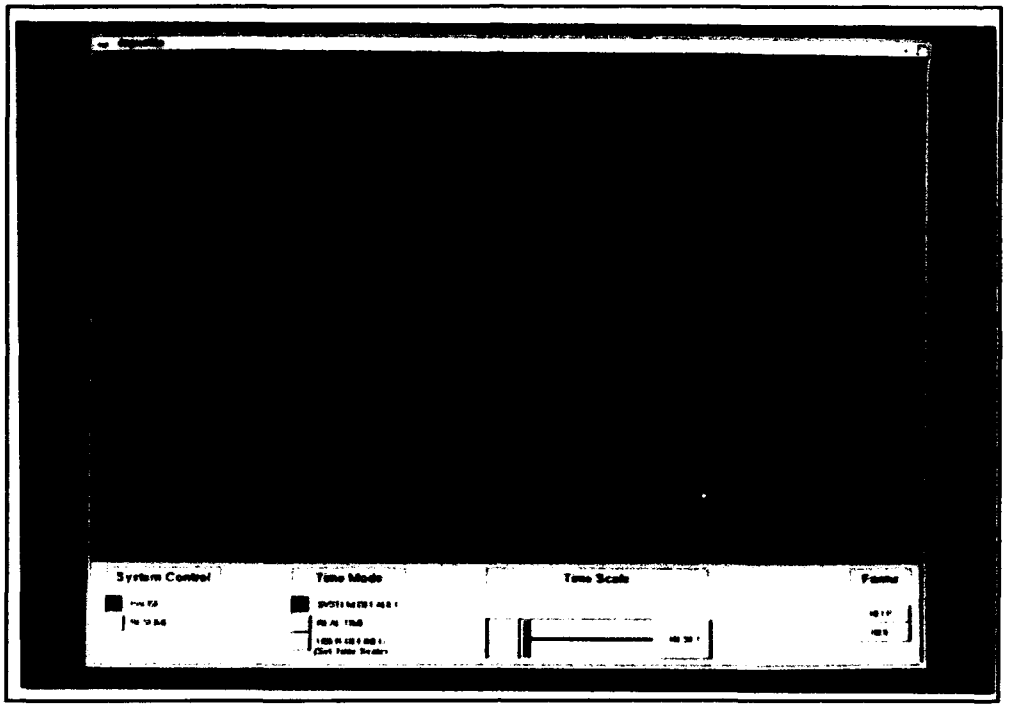


Figure 5.5 Visual Features
Satellite Locator and Mission Status Display

5.4 Summary

The Satellite Modeler supports dynamic configuration of visual features to analyze the behavior of satellites in near-earth orbit. Visually realistic satellite motion is the result of applying sophisticated computer graphics and rendering techniques to an established mathematical model for satellite orbit propagation. Similar methods achieve realism for the other members of the space environment itself.

A comparison of the Satellite Modeler to selected applications described in Chapter 2 reveals the strengths of the system as well as areas where functionality may be expanded. SM provides many of the same features and capabilities as VISIM [Mar93] including a graphical user interface, earth model display, file-based configuration, and spacecraft simulation. VISIM models extensive functionality related to on-board sensors and their fields of view. It also supports interactive

manipulation of satellite orientation and orientation by simulating thruster-firing activities. Both systems accept actual satellite telemetry for propagating models in the simulation. Models used by VISIM possess minimal detail whereas SM uses multiple resolutions of satellite models for visual realism. SM also supports a wide variety of configurable simulation views that, as yet, VISIM has not implemented such as a viewpoint attached to a satellite object or "flying" in its own orbit.

Satellite Modeler compares favorably with the systems described in [Oca90] and [Hag86]. The CGPP system, like SM, features a highly interactive and configurable user interface, a three-dimensional space environment, and real-time display of satellite orbital trails. CGPP is primarily menu-driven but supports push buttons. The CGPP user interface provides configuration capabilities similar to SM like selection of environment features and display of satellite orbital element set data.

Ocampo's system is to serve as an analysis and visualization tool for astronautics education. It employs not only an ECI-based three-dimensional model, but also supports a perifocal coordinate system for two-dimensional modeling. The earth is rendered in wire frame; satellite models are represented as points along the orbital path. The primary purpose of Ocampo's system is to provide visualization of orbital mechanics problems as a classroom tool. Students are able to see the results of orbital equations based on input parameters. Its emphasis is on a rigorous approach to mathematical modeling rather than visual realism, as is the case with Satellite Modeler.

The current focus of Satellite Modeler is to provide a visually realistic representation of the near-earth space and accurately model the orbital behavior of satellite objects in that environment. SM offers two capabilities that none of the three systems cited above currently deliver. First, it is designed to function as an immersive virtual environment that allows the user to interact directly with the simulation. To achieve this interaction, SM supports a variety of interface devices and allows the user to tailor each simulation to his or her preferences. And second, SM can function as a player in a real-time, interactive distributed simulation. Broadcasting satellite data across a network enhances the effectiveness of these simulations by providing a real-world capability.

VI. Thesis Summary

6.1 Introduction

Creating a virtual space environment to support satellite modeling and orbital analysis was an incremental process. A study of current work in supporting fields, summarized in Chapter 2, provided a conceptual foundation for determining the overall feasibility of this problem. Reviewing basic orbital mechanics principles led to an understanding of the approach others had taken in implementing similar systems. Chapter 3 provides a detailed description of the Satellite Modeler system design. This design was based on an object-oriented methodology and software engineering principles to ensure system extensibility and maintainability. Chapter 4 discusses the overall process of implementing this design, focusing on key system components, to achieve the results described in Chapter 5.

6.2 Recommendations for Future Work

The result of this thesis effort is a configurable, interactive tool for modeling and analysis of satellite orbits. While the features implemented in this version of the Satellite Modeler fulfill all of the established general requirements, many specific items were not incorporated due to time constraints and resource limitations. Where possible, templates have been constructed as place holders for functionality to be added during follow-on research. Table 6.1 is a snapshot of the work proposed in this chapter. Each of the primary system components -- user interface, space environment, satellite orbital model, and simulation view -- are candidates for future work, particularly with respect to accuracy and flexibility.

6.2.1 User Interface. In addition to the items listed below, two measures are needed to determine potential modifications to the user interface. First, the system should be placed at a user site for functional testing. Ease of use, appearance, and accessibility of required functions are the three of the metrics by which the user interface should be evaluated. Independent of functional testing, a user study should be conducted to research potential uses for the system by

Table 6.1
Proposed System Modifications

Functional Area	Proposed Work
User Interface	Conduct user study and perform functional testing at user site.
	Expand on-line help facility to operate in hypertext mode and/or support interactive query.
	Add interactive support for level-of-detail management.
	Provide interactive capability to add or remove configured constellations and satellites.
	Replace Interactive interface Forms screens with transparent interface objects.
	Explore input/output device support for stand-alone and virtual environments.
Space Environment	Model astrophysical phenomena (solar wind, radiation, magnetic fields, etc.)
	Analyze and incorporate shadow algorithm to achieve accurate lighting model.
	Decompose starfield into component constellations with appropriate magnitude, color, and scaled distance from the earth.
	Obtain and apply realistic texture map to moon.
	Improve accuracy of lunar propagation model.
	Determine desirability of modeling atmospheric conditions for earth model.
Satellite Orbital Model	Add collision detection algorithm.
	Provide for interactive manipulation of orbital elements.
	Process multiple element sets to simulate orbital transitions.
	Enhance accuracy of satellite model by applying rigid body dynamics.
	Model satellite sensor capabilities to determine FOV.
	Process actual satellite orientation data.
	Incorporate better estimate for satellite look angle as attribute of Satellite object class.
	Provide for manipulation of connected components subject to physical constraints.
Distributed Simulation	Add PDU "receive" capability.
Simulation Viewpoint	Restructure View object class to support interactive switching between multiple concurrent viewpoints.
	Adjust view position by dynamically calculating rate of change with respect to focus.

other Air Force or Department of Defense agencies. Existing system functionality should be demonstrated so that the feedback received can be incorporated into future baselines.

Proposed enhancements of the current SM user interface are as follows. The on-line help facility should be expanded to operate in a hypertext mode. This would result in a more interactive help dialogue by allowing the user to select certain objects or textual strings and receive expanded help information displayed in a pop-up window. An option should be added to satellite visual features whereby the user may select activation or deactivation of LOD management. Start-Up and Interactive interfaces need to incorporate options for adding and removing both entire constellations and individual satellites from the current simulation.

Support for Input/Output devices must be expanded. The interface currently provides selection of various devices, including an HMD, BioMuse, and Voice Navigation System (VNS). The VNS was not explored based on the security measures involved at the user site restricting speaker/microphone installation. The AFIT HMD did not provide adequate resolution for immersive visualization and the new BioMuse arrived too late in the present thesis cycle to be incorporated. A haptic device will be necessary for future implementations, especially to support interaction with the virtual space environment. This device might also be used with a virtual menuing system, such as the ones described in ([Jac92], [Wei89]), that could provide the same interface options while the user is immersed in the simulation environment.

Alternate methods of presenting interface objects should be explored. Use of the transparent "virtual" button objects defined by [Wil93] might provide continuous support for frequently performed tasks. These objects would overlay but not obstruct the simulation view. Considering current system performance, both Forms and GL functions operate concurrently without apparent conflict. The "virtual" buttons were proven in [Wil93] to offer substantial performance gains over Forms, especially while rendering an interactive simulation. Use of these transparent interface objects would prevent the simulation scene from freezing as it currently does whenever Interactive interface objects are activated.

6.2.2 Space Environment. The current space environment represents a first step. Visualizing different physical phenomena such as solar wind, radiation, and magnetic fields would contribute significantly to the analysis of satellite behavior. Further research into shadow algorithms may produce a method for modeling more accurate lighting, object shading, and shadows within the environment. The capability to determine when an orbiting spacecraft passes in and out of the earth's shadow exists. This can be done using a simple intersection calculation. A more complex lighting model is needed to control the quality of light in the environment and to quantify the amount of energy absorbed by a satellite's solar panels.

The environment starfield is also a candidate for improvement. New Performer capabilities allow for separating light point data types into clusters with individual distance, color, size, and intensity values rather than the single grouping currently implemented. By taking advantage of these new functions, stars may be grouped according to actual constellations for an even more realistic visual effect.

If atmospheric modeling for the earth is desired to visualize changes in cloud cover or environmental conditions, the approach taken by [Nis93] might be considered. This model produces a photorealistic earth with atmosphere and cloud cover. However, the computations involved in generating the model may not be worth the expense in processing time needed to render a constantly rotating earth. Regardless of the whether or not this option is explored, the earth model itself consumes a large percentage of the system's rendering time. The movement of satellite objects becomes noticeably slower as the earth appears in the simulation view. Level-of-detail switching applied to the earth model may alleviate this problem.

Finally, the lunar object should be improved with respect to visual realism and orbital accuracy. There was no texture map of the lunar surface available at the time it was introduced to the simulation; a texture pattern that most closely resembled the moon was selected. It should be a fairly straightforward task to scan a section of the lunar surface from an atlas and transfer the scanned image into a texture map file for application to the lunar sphere. The current simplified approach to propagating the moon should be modified to account for the perturbative forces that impact its orbit around the earth.

6.2.3 Satellite Orbital Model. Several areas remain to be addressed regarding the modeling of satellites in near-earth orbit. First and foremost is the issue of collision detection. One of the principal requirements for the Satellite Modeler was to provide visualization of two satellites in opposing orbits engaged in a docking maneuver. To implement such a feature, the system would have to provide interactive maneuvering of satellite models. As each object approached the other, it would be necessary to activate onboard thrusters to reduce their velocity. The system

could estimate a proximal distance at which it would configure the individual satellites as one object with a new, single element set for propagating its orbit.

A more accurate orbital model may be achieved by adding the capabilities to process multiple element sets and to accept actual satellite attitude information. Multiple element sets would allow simulation of orbital transitions where on-orbit burns must occur. The current system is only capable of handling one element set for each satellite. Propagation for an indefinite period is derived from this single element set; this does not accurately model the long-term life of an orbiting spacecraft. Incorporating this change will accurately represent a satellite's orbit over time and account for orbital change maneuvers and station-keeping activities. The current nadir-pointing orientation should be replaced with actual satellite attitude data. This information could be processed in conjunction with the orbital element set and would provide an accurate representation of how the satellite should be oriented.

Application of rigid-body dynamics, along with an accurate representation of the physical qualities of the satellite model, should result in more realistic motion and response to interactive manipulation. An estimated value for a satellite's look-angle should be added to the Satellite object class definition. This attribute would provide a better representation of where the ground trace should be focused and allow for interactive manipulation of the satellite's FOV. Adding a sensor capability to the satellite model could also enhance the visualization of a satellite's FOV.

Another requirement not addressed by this research is the capability to manipulate individual connected components on the satellite such as antennae, solar panels, and radar. Again, rigid body dynamics would be used to define the physical characteristics of each identified component. The user could then interactively manipulate one of these components through a haptic interface. Visualizing the physical effects on satellite behavior when grasping and repositioning a solar array would be particularly beneficial to analysts. Methods would have to exist to resolve critical conditions where an attempt is made to manipulate a component

beyond its range of motion. In this event, the system may classify the component as either "broken" or simply stop processing additional movement past the established limit.

6.2.4 Distributed Simulation. A final consideration with respect to enhancing satellite performance within the simulation is the addition of a "receive" capability to complement the broadcast capability. At present, each satellite can only broadcast PDUs information packets that contain standard position, orientation, and velocity information. A "receive" capability could provide for addition of interactive command and control of satellites within a distributed simulation environment. It could also be used to identify satellite models as destroyed or damaged by space debris; LOD management could be employed to switch to a destroyed or damaged model to visualize such events.

6.2.5 Simulation Viewpoint. Proposed modifications to the simulation view consist primarily of restructuring the object class and providing greater flexibility for interactive adjustment. In its current form, the Satellite View player class is a single object. It would make more sense to separate the different types of view into individual sub-classes of the view player. This separation would allow switching between multiple, concurrent viewpoints.

Adjusting the position and orientation of any given viewpoint currently uses a constant factor to increment these values based on the view distance from the focal point. When the view distance exceeds a defined limit, the view position changes more rapidly than if the distance is relatively small. One way to resolve this problem would be to provide an interface object allowing the user to control the speed with which the view changes position or orientation. Another option is to modify the view adjustment algorithm to dynamically compute the rate with which the view changes based on distance from the view focus.

6.3 Conclusions

The Satellite Modeler provides satellite system analysts the capability to model satellite behavior within the framework of a three-dimensional, orbital environment using either an immersive or "through-the-window" virtual reality paradigm. The system's direct manipulation, graphical interface allows users to tailor simulation performance to their specifications. Extensive facilities are provided for flexible visualization in either paradigm. Visual representation of the environment and the objects within it is accomplished using a physically accurate orbital model. Additional realism and user immersion within the environment are achieved through selective display of visual characteristics. Satellite Modeler functionality was expanded over the original prototype by adding an interface that allows it to share its data with other players in a networked simulation. The end result is a multi-role simulation that provides flexible visualization and direct interaction with orbiting spacecraft in the context of a three-dimensional space environment.

Bibliography

[App92] Appino, P. Lewis, J. B., Koved, L., Ling, D. T., Rabenhorst, D. A., Codella, C. F. "An Architecture for Virtual Worlds," in *Presence - Teleoperators and Virtual Environments,* Eds. Thomas B. Sheridan and Thomas A. Furness III. Cambridge, MA: MIT Press, 1992.

[Auk92] Aukstakalnis, S., Blatner, D. *Silicon Mirage - The Art and Science of Virtual Reality.* Berkeley, CA: Peachpit Press, 1992.

[Bat71] Bate, R., Mueller, D., White, J. *Fundamentals of Astrodynamics.* New York: Dover Publications, 1971.

[BBN92] Bolt, Beranek and Newman, Inc. *The SIMNET Network and Protocols.* BBN Report no. 7627. 1992.

[Ber86] Bergman, L. D., Fuchs, H., Grant, E., Spach, S. "Image Rendering by Adaptive Refinement," *Proceedings of the SIGGRAPH 86 (Dallas, Texas, August 18-22, 1986),* in *Computer Graphics,* Proceedings, Annual Conference Series 1986, ACM SIGGRAPH, New York, 1986, pp. 29-34.

[Ber91] Bergamasco, D., DeMicheli, D. M., Parrini, G., Salsedo, F., Marchese, S. S.," in *Proceedings of the Fifth International Conference on Advanced Robotics (ICAR) - Robots in Unstructured Environments,* Pisa, Italy, pp. 163-167, 19-22 June 1991.

[Bis92] Bishop, G., et. al. "Research Directions in Virtual Environments," *Computer Graphics, 26:* 154-177 (August 1992).

[Bla93] Blau, B., Moshell, J. M., McDonald, B. "The DIS (Distributed Interactive Simulation) Protocols and their Application to Virtual Environments," *Proceedings of the Meckler Virtual Reality '93 Conference.* (May 1993).

[Bro90] Brooks, F. P. Jr., Ouh-Young, M., Batter, J. J., Kilpatrick, P. J. P. "Project GROPE - Haptic Displays for Scientific Visualization," *Computer Graphics, 24:* 177-185, (August 1990).

[Bro84] Brotman, L., Badler, N. "Generating Soft Shadows with a Depth Buffer Algorithm," *IEEE Computers Graphics and Applications, 4:* 5-12 (October 1984).

[Bry92] Bryson, S. "Survey of Virtual Environment Technologies and Techniques," *Proceedings of the SIGGRAPH 92 (Chicago, Illinois, July 26-31, 1992).* In *Implementation of Immersive Virtual Environments (Course Notes).* Proceedings, Annual Conference Series, 1992, ACM SIGGRAPH, New York, August 1992, Chapter 1.

[Bux86] Buxton, W. "There's More to Interaction Than Meets the Eye: Some Issues in Manual Input," in *User Centered System Design -- New Perspectives in Human-Computer Interaction.* Eds. Donald A. Norman, Stephen W. Draper. Hilldale, NJ: Lawrence Erlbaum Associates, 1986.

[Cor89] Corbi, T. A. "Program Understanding: Challenge for the 1990s," *IBM Systems Journal 294-306.* Volume 28 Number 2, 1989.

[Dan88] Danby, J. *Fundamentals of Celestial Mechanics.* Richmond, VA: Willmann-Bell, 1988.

[Dav88] Davies, J. K. *Satellite Astronomy: The Principles and Practice of Astronomy from Space.* Chichester, West Sussex, England: Ellis Horwood Limited, 1988.

[Ell92] Ellis, S. R. "The Design of Virtual Spaces and Virtual Environments," in *Human Vision, Visual Processing, and Digital Display III,* Bernice E. Rogowitz, Editor, Proc. SPIE, 1666, pp. 536-540, (1992).

[Eri93] Erichsen, M. "Weapon System Sensor Integration for a DIS-Compatible Virtual Cockpit." MS thesis, AFIT/GCS/ENG/93D-07. Graduate School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1993.

[Eyl87] Eyles, D. E. "A Computer Graphics System for Visualizing Spacecraft in Orbit," *Proceedings of 1987 Conference on Spatial Displays and Spatial Instruments.* Chapter 36. Moffett Field, California, 1987.

[Fen91] Feng, X., Niederjohn, R. J., McGreevy, M.W. "An Intelligent Control and Virtual Display System for Evolutionary Space Station Workstation Design," *Proceedings of the Fifth Annual Workshop on Space Operations Applications and Research* (SOAR). 582-587. 1991.

[Fis93] Fisher, S. S. "Virtual Interface Environments," in *The Art of Human-Computer Interface Design,* Ed. Brenda Laurel. Reading, MA: Addison-Wesley, 1993.

[Fol90] Foley, J. D., van Dam, A., Feiner, S. K., Hughes, J. F. *Computer Graphics - Principles and Practice* (Second Edition). Reading, MA: Addison-Wesley, 1990.

[Gar93a] Gardner, B. "The Creator's Toolbox," in *Virtual Reality - Applications and Explorations.* Ed. Alan Wexelblat. Boston, MA: Academic Press Professional, 1993.

[Gar93b] Gardner, M. "A Distributed Interactive Simulation Based Remote Debriefing Tool for Red Flag Missions." MS thesis, AFIT/GCS/ENG/93D-09. Graduate School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1993.

[Gou85] Gould, J. D., Lewis, C. "Designing for Usability: Key Principles and What Designers Think," *Communications of the ACM, 28:* 300-311 (March 1985).

[Gig93] Gigante, M. A. "Virtual Reality: Enabling Technologies," in *Virtual Reality Systems*, Eds. R. A. Earnshaw, M. A. Gigante, H. Jones, London, England: Academic Press, 1993.

[Gri93] Grimsdale, C. "SuperVision - A Parallel Architecture for Virtual Reality," in *Virtual Reality Systems*, Eds. R. A. Earnshaw, M. A. Gigante, H. Jones, London, England: Academic Press, 1993.

[Hag86] Hagedorn, J., Ehrner, M., Reese, J., Chang, K., Tseng, I. "A Computer Graphics Pilot Project: Spacecraft Mission Support with an Interactive Graphics Workstation," *Proceedings of the SCS Simulators Conference 1986.* 61-64. San Diego, CA: Society for Computing Simulation, 1986.

[Hal87] Hallet, H., Jahnke, R. "Space Simulation Using Computer Generated Imagery," *Proceedings of the Conference on Aerospace Behavioral Technology.* 199-206. Warrendale, PA: Society of Automotive Engineers, Inc., 1988.

[Har91] Harvey, E. P., Schaffer, R. L. "The Capability of the Distributed Interactive Simulation Network Standard to Support High Fidelity Aircraft Simulation," *Proceedings of the Thirteenth Interservice/Industry Training Systems Conference.* 127-135. Orlando, Florida, 1991.

[Hec86] Heckbert, P. S. "Survey of Texture Mapping," *IEEE Computers Graphics and Applications, 6:* 56-67 (November 1986).

[Hit92] Hitchner, L. "Virtual Planetary Exploration: A Very Large Virtual Environment," *SIGGRAPH '92 Workshop on Implementation of Immersive Virtual Environments.* Chapter 6. New York: ACM Press, 1992.

[Hoo80] Hoots, F., Roehrich, R. L. "Models for Propagation of NORAD Element Sets," *Space Track Report Number 3*, Peterson AFB, Colorado, 1980.

[Iwa90] Iwata, H. "Artificial Reality with Force Feedback: Development of Desktop Virtual Space with Compact Master Manipulator," *Computer Graphics, 24:* 165-170 (August 1990).

[Jac86] Jacob, R. J. "A Simplification Language for Direct-Manipulation User Interfaces," *ACM Transactions on Graphics, 5:* 288-317 (October 1986).

[Jac92] Jacoby, R. "Design and Implementation Issues in the VIEW Lab," *Proceedings of the SIGGRAPH 92 (Chicago, Illinois, July 26-31, 1992).* In *Implementation of Immersive Virtual Environments (Course Notes).* Proceedings, Annual Conference Series, 1992, ACM SIGGRAPH, New York, August 1992, Chapter 5.

[Kai89] Kaiser, M., Proffitt, D. R. "Perceptual Issues in Scientific Visualization," *Three-Dimensional Visualization and Display Technologies,* Woodrow E. Robbins, Scott S. Fisher, Editors, Proc. SPIE, 1083, pp. 205-211, (1989).

[Kal93] Kalawsky, R. S. *The Science of Virtual Reality and Virtual Environments.* Wokingham, England: Addison-Wesley, 1993.

[Kau90] Kaufman, A., Yagel, R., Bakalash, R. "Direct Interaction with a 3D Volumetric Envirnoment," *Proceedings of the SIGGRAPH 90 (Dallas, Texas, August 6-10, 1990),* in *Computer Graphics*, Volume 24, Number 4 (August 1990), roceedings, Annual Conference Series, 1990, ACM SIGGRAPH, New York, 1990, pp. 33-34.

[Kob73] Kobayashi, H., Yamamoto, Y., Oka, M., Goma, K. "Interactive Graphic Display of Satellite Orbital Information," *Proceedings of the Tenth International Symposium on Space Technology and Science*. 901-908. Tokyo, Japan, 1973.

[Kur93] Kurtenbach, G., Hulteen, E. A. "Gestures in Human-Computer Communication," in *The Art of Human-Computer Interface Design,* Ed. Brenda Laurel. Reading, MA: Addison-Wesley, 1993.

[Led87] Lederman, S. "Hand Movements: A Window Into Haptic Object Recognition," *Cognitive Psychology, 19:* 342-368 (1987).

[Leh91] Lehnert, H., Blauert, J. "Virtual Auditory Environment," *Proceedings of the Fifth International Conference on Advanced Robotics (ICAR) - Robots in Unstructured Environments,* Pisa, Italy, 211-218, 19-22 June 1991.

[Lew91] Lewis, J. B., Koved, L., Ling, D. T. "Dialogue Structures for Virtual Worlds," *Proceedings of the ACM SIGGCHI CHI '91 (New Orleans, Louisiana, April 27-May 2, 1991).* 131-135. New York: ACM Press, 1991.

[Lin91] Ling, D. T. "Virtual Worlds and Teleoperations," *Proceedings of the Fifth International Conference on Advanced Robotics (ICAR) - Robots in Unstructured Environments,* Pisa, Italy, pp. 174-179, 19-22 June 1991.

[Log92] Logsdon, T. *The Navstar Global Positioning System.* New York: Van Nostrand Reinhold, 1992.

[Mar89] Marshall, G. "Back From the Past: The Helmet Integrated System of Albert Bacon Pratt," in *Helmet Mounted Displays*, Jerome T. Carollo, Editor, Proc. SPIE 1116, pp. 2-11, (1989).

[Mar93] Mara, S. "VISIM," *Journal of the British Interplanetary Society, 46:* 203-208, (1993).

[Mas92] Massimino, M., Sheridan, T. "Using Auditory and Tactile Displays for Force Feedback," *Telemanipulator Technology,* Hari Das, Editor, Proc. SPIE 1833, pp. 325-336, 1992.

[McD90] McDonald, L.B., Pinon, C., Glasgow, R., Danisas, K. "The Standardization of Protocol Data Units for Interoperability of Defense Simulations," *Proceedings of the Twelfth Interservice/Industry Training Systems Conference,* Orlando, Florida, pp. 93-102, 6-8 November 1990.

[McD91] McDonald, L. B., Bouwens, C. P, Hofer, R., Wiehagen, G., Danisas, K., Shiflett, J. "Standard Protocol Data Units for Entity Information and Interaction in a Distributed Interactive Simulation," *Proceedings of the Thirteenth Interservice/Industry Training Systems Conference,* Orlando, Florida, pp. 119-126, 1991.

[McG93] McGreevey, M. "Virtual Reality and Planetary Exploration," in *Virtual Reality -- Applications and Explorations.* Ed. Alan Wexelblat. New York: Academic Press, 1993.

[Mil88] Miller, D. C., Pope, A. R., Waters, R. M. "Long-haul Networking of Simulators," *Proceedings of the Eighth Interservice/Industry Training Systems Conference,* Orlando, Florida, pp. 577-582, 1988.

[Min90] Minsky, M., Ouh-Young, M., Steele, O., Brooks, F. P., Jr., Behensky, M. "Feeling and Seeing: Issues in Force Display," *Computer Graphics, 24:* 235-242 (1990).

[Mou93a] Mountford, S. J., Gaver, W. W. "Talking and Listening to Computers," in *The Art of Human-Computer Interface Design,* Ed. Brenda Laurel. Reading, MA: Addison-Wesley, 1993.

[Mou93b] Mountford, S. J. "Tools and Techniques for Creative Design," in *The Art of Human-Computer Interface Design,* Ed. Brenda Laurel. Reading, MA: Addison-Wesley, 1993.

[Nai76] Naimark, M. "Elements of Realspace Imaging: A Proposed Taxonomy," *Stereoscopic Displays and Applications II,* John O. Merritt, Scott S. Fisher, Editors, Proc. SPIE 1457, pp. 169-179, (1991).

[Neg93] Negroponte, N. "Hospital Corners," in *The Art of Human-Computer Interface Design,* Ed. Brenda Laurel. Reading, MA: Addison-Wesley, 1993.

[Nis93] Nishita, T., Sirai, T., Tadamura, K., Nakamae, E. "Display of the Earth Taking into Account Atmospheric Scattering," *Proceedings of the SIGGRAPH 93 (Anaheim, California, August 1-6, 1993).* In *Computer Graphics* Proceedings, Annual Conference Series, 1993, ACM SIGGRAPH, New York, 1993, pp. 175-182.

[Oca90] Ocampo, C. "Computer Graphics Applied to the Visualization of Two-Body Orbital Mechanics Problems," *Proceedings of the 28th Aerospace Sciences Meeting*, Reno, Nevada, 1990.

[Pas92] Pasachoff, J. M., Menzel, D. H. *Stars and Planets* (Third Edition). Boston, MA: Houghton-Mifflin, Inc., 1992.

[Pau91] Pausch, R., Dwivedi, P., Long, A. "A Practical, Low-Cost Stereo Head-Mounted Display," *Stereoscopic Displays and Applications II*, John O. Merritt, Scott S. Fisher, Editors, Proc. SPIE 1457, pp.198-208 , (1991).

[Pea85] Peachey, D. R. "Solid Texturing of Complex Surfaces," *Proceedings of the SIGGRAPH 85 (San Francisco, July 22-26, 1985)*, in *Computer Graphics* Proceedings, Annual Conference Series, 1985, ACM SIGGRAPH, New York, 1985, 279-285.

[Pim93] Pimentel, K., Teixeira, K. *Virtual Reality: Through the New Looking Glass.* New York: McGraw-Hill, 1993.

[Pon92] Pond, D. "A Synthetic Environment for Satellite Modeling and Satellite Orbital Motion," *MS Thesis*, AFIT/GCS/ENG/92D-12, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1992.

[Pou92] Poulin, P., Fournier, A. "Lights from Highlights and Shadows," *Proceedings of the 1992 Symposium on Interactive 3D Graphics*. 31-38. New York: ACM Press, 1992.

[Pyl91] Pyle, I. C. "Programming for Safety," in *Developing Safety Systems*, Chapter 4, New York: Prentice Hall, 1991.

[Qua90] Quam, D. L. "Gesture Recognition with a DataGlove," *Proceedings of the IEEE National Association of Engineering Conference (NAECON)*. 755-760. Washington, D.C.: IEEE Computer Society Press, 1990.

[Reb89] Rebo, R. K., Amburn, P. "A Helmet-Mounted Virtual Environment Display System," *Helmet Mounted Displays*, Jerome T. Carollo, Editor, Proc. SPIE 1116, pp. 80-84, (1989).

[Rim89] Rimrott, F. *Introductory Orbit Dynamics*. Braunschweig, Germany: Frieder Vieweg & Son, 1989.

[Roy91] Roy, A. E. *Orbital Motion* (Third Edition). Bristol, England: Adam Hilger, 1991.

[Sal93] Salomon, G. "New Uses for Color," in *The Art of Human-Computer Interface Design*, Ed. Brenda Laurel. Reading, MA: Addison-Wesley, 1993.

[Sch90a] Schiffman, H. R. *Sensation and Perception: An Integrated Approach* (Third Edition). New York: John Wiley & Sons, 1990.

[Sch93] Schmandt, C. "Illusion in the Interface," in *The Art of Human-Computer Interface Design,* Ed. Brenda Laurel. Reading, MA: Addison-Wesley, 1993.

[Sch90b] Schroder, P., Zeltzer, D. "The Virtual Erector Set: Dynamic Simulation with Linear Recursive Constraint Propagation," *Proceedings of the 1990 ACM Symposium on Interactive 3D Graphics.* 23-31. New York: ACM Press, 1990.

[Sha92] Shaw, C., Liang, J., Green, M., Sun, Y. "The Decoupled Simulation Model for Virtual Reality Systems," *Human Factors in Computing Systems - Proceedings of the 1992 ACM SIGCHI CHI '92 (Monterey, California, May 3-7, 1992).* 321-328. New York: ACM Press, 1992.

[Sil87] Silverstein, L. "Human Factors for Color Display Systems: Concepts, Methods, and Research," in *Color and the Computer.* 27-61. New York: Academic Press, 1987.

[Smi83] Smith, R. S., Davis, P., Goodwin, D. "Dynamic Spacecraft Simulators in Military Space Ground Systems," *Proceedings of the Symposium on Military Space Communications and Operations*, USAF Academy, Colorado, pp. 119-126, 1983.

[Sny93] Snyder, M. "ObjectSim - A Reusable Object Oriented DIS Visual Simulation," *MS Thesis,* AFIT/GCS/ENG/93D-20, Graduate School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1993.

[Spe92] Speeter, T. H. "Transforming Human Hand Motion for Telemanipulation," in *Presence - Teleoperators and Virtual Environments.* 63-79. Eds. Thomas B. Sheridan and Thomas A. Furness III. Cambridge, MA: MIT Press, 1992.

[Sto91] Stone, R. J. "Advanced Human-System Interfaces for Telerobotics Using Virtual Reality & Telepresence Technologies," in *Proceedings of the Fifth International Conference on Advanced Robotics (ICAR) - Robots in Unstructured Environments,* Pisa, Italy, pp. 168-173, 19-22 June 1991.

[Str92] Strasser, A. "Development of High Resolution Head-Ported Displays for a Virtual Environment," *Display Technologies,* Shu-Hsia Chen, Shin-Tson Wu, Editors, Proc. SPIE 1815, pp. 188-193, (1992).

[Sut68] Sutherland, I. E. "A Head-Mounted Three-Dimensional.Display," in *Proceedings of the Fall Joint Computer Conference*, Washington D.C.: Thompson Books, pp 757-764, 1968.

[Tho88] Thorpe, J. "Warfighting with SIMNET - A Report From the Front," *Proceedings of the 10th Interservice/Industry Training Systems Conference*, Orlando, Florida, pp. 263-273, 1988.

[Ver93] Vertelney, L., Arent, M. "Two Disciplines in Search of an Interface," in *The Art of Human-Computer Interface Design.* Ed. Brenda Laurel. Reading, MA: Addison-Wesley, 1993.

[Vin92] Vince, J. *3-D Computer Animation.* Wokingham, England: Addison-Wesley, 1992.

[Wal93] Walker, J. "Through the Looking Glass," in *The Art of Human-Computer Interface Design,* Ed. Brenda Laurel. Reading, MA: Addison-Wesley, 1993.

[Wan90] Wang, J., Azuma, R., Bishop, G., Chi, V., Eyles, J., Fuchs, H. "Tracking a Head-Mounted Display in a Room-Sized Environment with Head-Mounted Cameras," *Helmet-Mounted Displays II,* Ronald J. Lewandowski, Editor, Proc. SPIE 1290, pp. 47-59, (1990).

[War92] Ward, M., Azuma, R., Bennett, R., Gottschalk, S., Fuchs, H. "A Demonstrated Optical Tracker With Scalable Work Area for Head-Mounted Display Systems," *Proceedings of the 1992 ACM Symposium on Interactive 3D Graphics.* 43-52. New York: ACM Press, 1992.

[War88] Ware, C., Jessome, D. R. "Using the Bat: A Six-Dimensional Mouse for Object Placement," *IEEE Computers Graphics and Applications, 8*: 65-70 (November 1988).

[Wat92] Watt, A., Watt, M. *Advanced Animation and Rendering Techniques.* New York: ACM Press, 1992.

[Wei89] Weimer, D., Ganapathy, S. K. "A Synthetic Visual Environment with Hand Gesturing and Voice Input," *ACM SIGGCHI Bulletin, 20:* 235-240 (1989).

[Wex93] Wexelblat, A. "The Reality of Cooperation: Virtual Reality and CSW," in *Virtual Reality - Applications and Explorations.* Ed. Alan Wexelblat. Boston, MA: Academic Press Professional, 1993.

[Wic93] Wick, D. T., Shehad, N. M., Hajare, A. R. "Testing the human computer interface for the telerobotic assembly of the Space Station," *Proceedings of the Fifth International Conference on Human-Computer Interaction (HCI International '93),* Orlando, Florida, pp. 213-218, 1993.

[Wil93] Wilson, K. G. "Synthetic Battle Bridge: Information Visualization and User Interface Design Applications in a Large Virtual Reality Environment," *MS Thesis,* AFIT/GCS/ENG/93D-26, Graduate School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, Ohio, December 1993.

[Woo90] Woo, A., Poulin, P., Fournier, A. "A Survey of Shadow Algorithms," *IEEE Computers Graphics and Applications:* 13-31 (1990).

[Wys93] Wyshynski, S., Vincent, V. J. "Full-Body Unencumbered Immersion in Virtual Worlds (The Vivid Approach and the Mandala® VR System," in *Virtual Reality - Applications and Explorations.* Ed. Alan Wexelblat. Boston, MA: Academic Press Professional, 1993.

[Zei92] Zeilek, M., Gregory, S. A., Elske, v. P. S. *Introductory Astronomy and Astrophysics.* Fort Worth: Saunders College Publishing, 1992.

[Zyd92] Zyda, M. J., Pratt, D. R., Monahan, J. G., Wilson, K. P. "NPSNET: Constructing a 3D Virtual World," *Proceedings of the 1992 Symposium on Interactive 3D Graphics.* 147-156. New York: ACM Press, 1992.

External Documentation

The following documents are maintained as part of the Satellite Modeler baseline. They are located in the AFIT Graphics Lab, Room 2011:

Satellite Modeler User's Manual

Satellite Modeler Programmer's Manual

Satellite Modeler Program Listing

Reference Material

The following materials were referenced throughout the development of this thesis:

AU-18, Space Handbook, 1985

CRC Standard Mathematical Tables, 29th Edition, 1987

Fundamentals of Physics by Halliday and Resnick, Third Edition, 1988

Iris Performer Programmer's Manual

Iris Performer Man Pages

Principles of Object-Oriented Analysis and Design by James Martin, 1993

Vita

Captain Andrea A. Kunz was born on 31 July 1961 in Heidelberg, West Germany. After being imported to the great state of Texas and growing up pretty much in one place -- Fort Worth -- she graduated from Southwest High School in 1979. She attended Texas Christian University, also in Fort Worth, graduating with a Bachelor of Arts in Computer Science in May 1983. After commissioning and graduation, she was assigned to Offutt AFB, Nebraska as a Commanding and Telemetry Software Analyst for the 1000th Satellite Operations Group, Air Force Space Command. She was responsible for maintaining the integrity of the Stored Telemetry Processing Subsystem for the Defense Meteorological Satellite Program (DMSP). In 1986, she received her first Air Force Institute of Technology assignment to an Education With Industry (EWI) position with Hughes Aircraft Company in El Segundo, California, where she worked in the Artificial Intelligence Technology Department of the Electro-Optical and Data Systems Group. Upon completion of her EWI assignment, she was assigned back to her home state, to the Air Force Military Personnel Center at Randolph AFB. There, she led a team of computer software engineers and personnel specialists in the maintenance of the Procurement Management Information System (PROMIS), used by the United States Air Force Recruiting Service to attract and retain qualified Air Force airmen and officers. She remained in that position until her selection to attend the Graduate School of Engineering, Air Force Institute of Technology, in May 1992.

Permanent address: 3717 Winifred Drive
Fort Worth, Texas 76133

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1993	Master's Thesis

4. TITLE AND SUBTITLE
A VIRTUAL ENVIRONMENT FOR SATELLITE MODELING AND ORBITAL ANALYSIS IN A DISTRIBUTED INTERACTIVE SIMULATION

5. FUNDING NUMBERS

6. AUTHOR(S)
Andrea A. Kunz, Capt, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCS/ENG/93D-14

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Joint National Intelligence Defense Staff (JNIDS)
4600 Silver Hill Road
Washington D.C., 20389

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

In response to the need for a more realistic, interactive, and immersive space simulation system, the Joint National Intelligence Defense Staff (JNIDS) has sponsored the development of the Satellite Modeler at the Air Force Institute of Technology (AFIT). The Satellite Modeler (SM) is a virtual environment (VE) application that allows analyst-users to enter a virtual near-earth space environment and visualize realistic satellite models performing accurate orbital motion. The Satellite Modeler provides manipulation functions that allow a user to interact with multiple satellite models and satellite constellations. The system affords the user multiple vantage points within the environment to view satellites in orbit. It also functions as a network actor in a distributed simulation environment. The Satellite Modeler achieves accurate physical modeling of satellite motion by using the North American Air Defense (NORAD) Command SGP4 orbital model and associated orbital elements for satellites currently in orbit. System functionality is realized within an object- oriented framework and accessible through a graphical user interface (GUI). This thesis is the second of a three-year effort to create a three-dimensional virtual environment for modeling and manipulating satellite objects.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Virtual Environments, Orbital Mechanics, Distributed Simulation, Object-Oriented, Computer Graphics, Graphical User Interface	97
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL